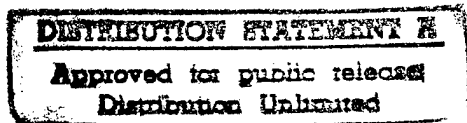


Research and Design of Intelligent Many-Agent Systems

Final Report for ONR grant no. N00014-94-1-0676

submitted by

Dr. John S. Bay
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute & State University
Blacksburg, Virginia 24061-0111



ABSTRACT

This report documents efforts under ONR grant no. N00014-94-1-0676. This is an AASERT award attached to parent grant NRL no. N00014-93-1-G022. The purpose of the grant is to support research on how group dynamics can emerge from collections of agents that would enable them to make decisions that individuals could not or accomplish tasks that individuals could not.

Funding from the grant supported four graduate students directly; i.e., with stipends and tuition, and a number of undergraduate students indirectly, through materials and supplies purchases to support their independent study efforts in distributed intelligence and cooperative robotics.

Results of these studies indicate that among distributed/cooperative learning methods, the most promising and appropriate for distributed mobile agent applications is a combination of learning and behavioral methods. In particular, the recommended method combines the data structures and execution cycle of the learning classifier system with reinforcement computed similarly to Q-learning and with some stochastic selection and genetics-based rule-parsing methods. These systems, in conjunction with message-based communications between agents, is shown to be widely applicable and convergent in ideal scenarios. The methods have the disadvantages of being slow, and they do not perform well in sequential learning tasks without significant modifications.

19971124 044

DTIC QUALITY INSPECTED 3

Overview

The purpose of the parent grant for this AASERT award was to examine methods for the coordination of intelligent agents by modeling them individually as dynamic subsystems. The problem spaces as a whole then represented a larger space into which the agents were embedded. The agents would couple to each other in ways that would result in the effective emergence of a single collective group dynamic which is, in a mathematically analytical manner, a composite system integrally dependent on each subsystem (agent). The agents' own perception of the coupling signals would therefore represent knowledge of the state of the population as a whole. The problem addressed by this research was the mechanism by which such coupling signals could be implemented, the information content of such signals, and the behaviors and decisions that could thereby be effected.

The original approach to this problem, as supported under the parent grant, was to encode signals and behavioral cues into the wave characteristics of coupled nonlinear oscillations. This model was inspired by biological systems, such as those that control circadian rhythms, locomotory systems, and learned responses to sensory stimuli. Because this parent grant was terminated prematurely in the first year, this original approach was broadened under the AASERT award to befit more general application domains, while still limiting the studies to distributed AI and decision theories for multiple agents.

Student Projects Supported

This AASERT award funded tuition and stipends for four graduate students, who together completed three M.S. degrees and is soon to result in one Ph.D., all in electrical engineering. In addition, the material and supply budget provided by the grant was used to purchase electronic components and software used by undergraduate independent study students in a supporting role. These students assisted the graduate students in hardware experiments, software simulation, and proof of concept projects. Such projects included the construction of radio-frequency coupling transceivers, small robotic platforms, and software modules for the implementation of the devised learning and coordination techniques.

Subsequent sections contain the thesis abstracts of each student. Following these are copies of conference and journal publications authored or co-authored by these students.

"Architecture Design and Simulation for Distributed Learning Classifier Systems," M.S. thesis by Douglas G. Gaff, 1995.

Abstract: In this thesis, we introduce the Distributed Learning Classifier system (DLCS) as a novel extension of J. H. Holland's standard learning classifier system. While the standard LCS offers effective real-time control and learning, one of its limitations is that it does not provide a mechanism for allowing communication between LCS agents in a multiple-agent scenario. Often multiple-agents are used to solve large tasks collectively by subdividing the task into smaller parts. Multiple agents can also be used to solve a task in parallel so that a solution can be arrived at more rapidly. With the DLCS, we introduce mechanisms that satisfy both of these cases, while still providing compatible operation with the LCS.

We introduce three types of messages that can be passed between DLCS agents. The first, the classifier message, allows agents to share learned information with one another, thereby helping agents benefit from each other's successes. The second, the action message, allows agents to "talk" to one another. The third, the bucket brigade algorithm payoff message, extends the chain rewarding payoff scheme of the standard LCS to multiple DLCS agents.

Finally, we present some simulation results for both the standard LCS and the DLCS. Our LCS simulations examine some of the important aspects of learning classifier system operation, as well as illustrate some of the shortcomings. The DLCS simulations justify the distributed architecture and suggest future directions for achieving learning among multiple agents.

[Comments: Mr. Gaff's research is the first known distributed system adaptation of the LCS. The resulting paper (see Appendix) was well-received. The results, though, served to illustrate limitations in the learning performance of the LCS that was to be addressed in the next M.S. thesis project to be supported. Mr. Gaff is currently a senior software engineering for Spatial Positioning Systems, Inc., of Reston, VA.]

"A Distributed Q-learning Classifier System for Task Decomposition in Real Robot Learning Problems," M.S. thesis by Kevin L. Chapman, 1996.

Abstract: A distributed reinforcement-learning system is designed and implemented on a mobile robot for the study of complex task decomposition in real robot learning environments. The distributed Q-learning Classifier System (DQLCS) is evolved from the standard Learning Classifier System (LCS) proposed by J. H. Holland. Two of the limitations of the standard LCS are its monolithic nature and its complex apportionment of credit scheme, the bucket brigade algorithm (BBA). The DQLCS addresses both of these problems as well as the inherent difficulties faced by learning systems operating in real environments.

We introduce Q-learning as the apportionment of credit component of the DQLCS, and we develop a distributed learning architecture to facilitate complex task decomposition. Based upon dynamic programming, the Q-learning update equation is derived and its advantages over the complex BBA are discussed. The distributed architecture is implemented to provide for faster learning by allowing the system to effectively decrease the size of the problem space it must explore.

Holistic and monolithic shaping approaches are used to distribute reward among the learning modules of the DQLCS in a variety of real robot learning environments. The results of these experiments support the DQLCS as a useful reinforcement learning paradigm and suggest future areas of study in distributed learning systems.

[Comments: Though not fully reflected in the abstract, the primary contributions of Mr. Chapman's thesis are in the decomposition of sequential learning tasks and their implementation on robotic hardware. It is in robot learning and sensor fusion that a novel application soon presented itself, although the solutions methods were to eventually differ (see below). Mr. Chapman is now a software engineer for the Intelligent Decision Support Systems Group at Raytheon E-Systems Corporation in Falls Church Virginia.]

"A Fuzzy Logic Solution for Navigation of the Subsurface Explorer Planetary Exploration Robot," M.S. thesis by Veronica A. Gauss, 1997.

Abstract: An unsupervised fuzzy logic navigation algorithm is designed and implemented in simulation for the Subsurface Explorer planetary exploration robot. The robot is intended for the subterranean exploration of Mars, and will be equipped with acoustic sensing for detecting obstacles. Measurements of obstacle distance and direction are anticipated to be imprecise however, since the performance of acoustic sensors is degraded in underground environments. Fuzzy logic is a satisfactory means of addressing imprecision in plant characteristics, and has been implemented in a variety of autonomous vehicle navigation applications. However, most fuzzy logic algorithms that perform well in unknown environments have large rule-bases or use complex methods for tuning fuzzy membership functions and rules. These qualities make them too computationally intensive to be used for planetary exploration robots like SSX.

In this thesis, we introduce an unsupervised fuzzy logic algorithm that can determine a trajectory for the SSX through unknown environments. This algorithm uses a combination of simple fusion of robot behaviors and self-tuning membership functions to determine robot navigation without resorting to the degree of complexity of previous fuzzy logic algorithms.

Finally, we present some simulation results that demonstrate the practicality of our algorithm in navigating indifferent environments. The simulations justify the use of our fuzzy logic technique, and suggest future areas of research for fuzzy logic navigation algorithms.

[Comments: Ms. Gauss had previously worked for the NASA Jet Propulsion Laboratory in the Mars Pathfinder program. The subsurface explorer seemed a suitable platform for application of distributed learning methods, and also provided the possibility to leverage any successful results into future research. However, she was given complete freedom to explore and determine the most feasible and appropriate control method for the SSX, and based on significant comparative studies, chose the fuzzy controller described. Fuzzy controllers, though, show little promise for distributed applications. Ms. Gauss is now a software engineer at Computer Motion, Inc., of Goleta, CA. Computer Motion makes robots for surgical applications]

"Q-learning in a Production Rule System for Applications to Control Systems," tentative title for Ph.D. thesis by Paul J. Johnson, 1997.

Tentative Abstract: The Learning Classifier System (LCS), originally proposed by John Holland in 1986, combines credit assignment (*i.e.* reinforcement learning) and rule discovery into an adaptive, message-passing, production rule system (PRS) capable of learning how to both plan and react in response to sensory inputs. While the LCS shows great promise in principle, it has not been very successful in control system applications, most likely due to its method of credit assignment: the Bucket Brigade Algorithm.

To improve the performance of Holland's original LCS in control system applications, this thesis proposes the use of Q-learning as a replacement for the credit assignment component of the LCS. The resulting system, termed the Q-learning Production Rule System (QPRS), maintains a rule discovery mechanism to complement the reinforcement learning capabilities associated with Q-learning. The QPRS requires minor modifications to the LCS rule discovery

mechanism to account for the new reinforcement learning component. Likewise, the Q-learning algorithm has been modified slightly from its original form to function within the overall structure of the QPRS.

Q-learning is used within the QPRS as a reinforcement learning mechanism, but it is equally valid to consider Q-learning to be a form of recursive dynamic programming. In this sense, Q-learning can be susceptible to the same problems that plague dynamic programming. One such problem is the curse of dimensionality. As the name suggests, as the number of discrete state-action pairs increases, full enumeration of all state-action possibilities can require prohibitive computing resources. However, when Q-learning is combined with rule discovery in the QPRS, this curse of dimensionality can be attenuated. By providing the QPRS with the capabilities to discover previously untested state-action pairs, it is not necessary to fully enumerate the entire state-action space throughout all time.

When the states and/or actions are continuous variables, as is usually the case in control system problems, some form of quantization must first take place. The concept of full state enumeration has a slightly different meaning when the state-action pairs are quantized versions of continuous variables. It is up to the designer to choose the level of quantization. This level of quantization will directly affect the problems associated with the curse of dimensionality. There is a tradeoff between increasing the resolution (finer quantization) and making the problem more computationally intensive. This tradeoff leads directly to one of the biggest changes we have introduced to the QPRS. We have modified the Q-learning algorithm within the QPRS to incorporate interpolation in the Q-value update equations. We make use of our dynamic programming analogy to create a better PRS by extending its interpolation algorithm to learning systems. Interpolation allows for a finer degree of response from a given level of quantization. This eliminates the need to more finely quantize a given variable.

The ties between DP and control systems suggest meaningful cost functions and specific ways to change cost functions to elicit desired responses. This is not the case with the BBA, nor with other "ad hoc" cost functions. There is a long history behind DP, and we wish to use this experience to develop cost functions to use with Q-learning in our QPRS. The use of a modified form of Q-learning has provided the QPRS with the capabilities needed for successful application to control system problems. With the development of the QPRS, a new tool now exists for adaptive optimal control in the absence of plant/model information.

[Comments: This study focused on the predictive study of the underlying dynamics of classifier-like systems and systems with Q-learning-like reinforcement methods. It provides a more analytical framework for the study of learning system convergence. Paul Johnson has finished his research on this topic and has only to complete and defend his dissertation. He is now a research engineer for Raytheon Corporation in Massachusetts.]

Appendix

Attached are copies of papers authored or co-authored by students supported directly and indirectly by project funds, and papers for which the undergraduate students provided laboratory assistance. Additional papers will appear in the future.

A. Reactive Coordination Scheme for a Many-Robot System

Kimberly Sharman Evans, Cem Ünsal, *Member, IEEE*, and John S. Bay, *Senior Member, IEEE*

Abstract—This paper presents a novel approach for coordinating a homogeneous system of mobile robots using implicit communication in the form of broadcasts. The broadcast-based coordination scheme was developed for the Army Ant swarm—a system of small, relatively inexpensive mobile robots that can accomplish complex tasks by cooperating as a team. The primary drawback, however, of the Army Ant system is that the absence of a central supervisor poses difficulty in the coordination and control of the agents. Our coordination scheme provides a global “group dynamic” that controls the actions of each robot using only local interactions. Coordination of the swarm is achieved with signals we call “heartbeats.” Each agent broadcasts a unique heartbeat and responds to the collective behavior of all other heartbeats. We generate heartbeats with van der Pol oscillators. In this application, we use the known properties of coupled van der Pol oscillators to create predictable group behavior. Some of the properties and behaviors of coupled van der Pol oscillators are discussed in detail. We emphasize the use of this scheme to allow agents to simultaneously perform an action such as lifting, steering, or changing speed. The results of experiments performed on three actual heartbeat circuits are presented and the behavior of the realized system is compared to simulated results. We also demonstrate the application of the coordination scheme to global speed control.

I. INTRODUCTION

MOBILE robots are increasingly being considered for industrial, military, and scientific applications. Much of the robotics research to date has focused on improving the sophistication of individual robots to accomplish more demanding and complex tasks. We propose that many tasks, such as large material handling problems, are best achieved using large numbers of relatively unintelligent robots. Typically, a distributed approach is more desirable in such a scenario since the communications overhead necessary for central control is prohibitive for large systems of mobile robots, or *swarms*. We further suggest that homogeneous swarms, which are composed of similar robots, have many advantages over heterogeneous systems. The goal of the *Army Ant Project* [2],

[18], [19] is to develop a homogeneous system of autonomous mobile robots with the following characteristics:

- The Army Ant system is *modular*, so that agents are interchangeable. Since any agent can assume the role of any failed teammate, the system is immune to the single point failures that plague heterogeneous systems where a failure in the chain of command can result in system failure. The failure of one, or even several, army ants will not adversely affect the performance of the swarm. The innate *robustness* of the Army Ant swarm is perhaps the most critical result of homogeneity.
- The Army Ant swarm is more *dynamic* than a heterogeneous system in that it may divide into smaller groups when fewer agents are required for a task. There is never a risk of incomplete hierarchies. Furthermore, homogeneous systems, because they typically do not use centralized control methods, can accommodate large numbers of agents.
- Army ants are physically small and relatively unsophisticated. The simplicity of the ants, combined with the features discussed above, will allow them to be *mass produced* fairly inexpensively. The intention is to create a system that can tolerate the loss of a few robots, both in terms of cost and system performance. A possible drawback of homogeneity is the possibility of agents being overqualified for a job, but the robustness of the system more than compensates for this cost.

Army ant agents are completely autonomous and have no *a priori* information about their environment. Their behavior is reactive in that individual agents respond to stimuli in the form of sensory inputs. This type of behavior, known as *sensor driven behavior*, offers greater flexibility to cope with changing environments. Unlike centralized, planner based approaches, wherein robots expend resources gathering and processing information, a reactive approach allows robots to respond quickly to changes in the environment.

Sensor driven behavior is consistent with a distributed control approach. An agent's control algorithm determines an action based on the local information that flows from sensory inputs. Group behavior is achieved through the simple actions of many unintelligent agents. For example, an agent is attracted to a payload destination by receiving an infrared signal from a beacon that is placed at the proper location. Each agent responds independently to the beacon, but the actions of each individual result in the group of agents gathering at the beacon.

The flexible nature the Army Ant swarm lends itself to many different applications. Army ants are well suited for

Manuscript received January 2, 1995; revised February 14, 1996 and June 14, 1996. This paper is based upon work supported in part by the Naval Research Laboratory under Grant N00014-93-1-G022, Office of Naval Research under Grant N0014-94-1-0676, and the National Science Foundation under Grant IRI-9202423. The content of this paper does not necessarily reflect the position or policy of the United States Government.

K. Sharman Evans is with Digital Directory, Knoxville, TN 37923 USA.
C. Ünsal is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: unsal@ri.cmu.edu).

J. S. Bay is with the Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0111 USA (e-mail: bay@vt.edu).

Publisher Item Identifier S 1083-4419(97)03878-8.

hazardous operations such as mine sweeping, nuclear power plant maintenance work and military applications, where the environment is unsafe for humans, and the risk factor is too high to utilize expensive, highly specialized robots.

The Army Ant approach is not intended to be the answer to all autonomous robot applications. However, the fact that not all robotic tasks require highly intelligent agents and that a group of *self-organizing* agents can exhibit a higher-level behavior/intelligence emerging as a result of agent interactions, makes this approach attractive. It is obvious that Army Ant robots are not appropriate for tasks requiring specialized division of labor such as assembling a complex machine. Our approach is appropriate for teams of unskilled laborers with little differentiation of responsibilities. It is not a valid proposition where there is no economy of scale, where a few intelligent agents are far better than a swarm of "unintelligent" agents.

A. Cooperative Behavior

The ability of the Army Ant swarm to accomplish complex goals relies upon *implicit* cooperation between individual agents. It is essential to distinguish between *explicit* and *implicit* cooperation when describing the behavior of a system of mobile robots. Mataric [16] defines *explicit* cooperation as "a set of interactions between agents which involve exchanging information or performing actions in order to help other agents achieve their goals." In contrast, *implicit* cooperation consists of "actions that are a part of the agent's own goal-achieving behavior, although they may have effects in the world that help other agents achieve their goals."

We recognize *implicit* cooperation as the type of interaction found quite frequently in natural systems. Insects are not altruistic, yet colonies of insects collectively accomplish goals such as transporting food and building structures [8], [9], [12]. Various studies of insects have shown that colonies operate in a distributed fashion, where individual insects follow a few simple rules. Likewise, simple interactions between relatively unintelligent army ants will produce rather complex system behavior. Beni and Wang [4] refer to this phenomenon as *swarm intelligence*.

Although both explicit and implicit communications may complement each other, as suggested by several researchers [1], [12], use of explicit communications has its problems in a swarm of the size envisioned by our approach. Bandwidth issues is one, as well as the need to "address" the agents, which will cause the swarm to lose its homogeneity. Explicit communications also adds to cost and complexity. These are two important issues we want to avoid in our scenario. It has been shown that "decentralized control without explicit communication can be used in performing cooperative tasks requiring a collective behavior" [12].

There are many instances when agents must not just cooperate, but do so in a coordinated manner. While to *cooperate* means to work toward a common goal, to *coordinate* means to perform a common action or movement in a harmonized manner. In the material transport example, agents may have to lift a pallet simultaneously so that the payload stays level; also,

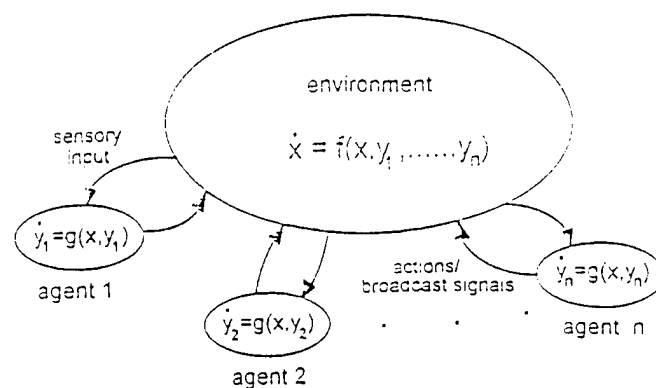


Fig. 1. Group dynamics in the Army Ant scenario.

transporting the payload is a much easier task when agents steer together. Various agent actions may, at some time or another, have to be performed synchronously. In this paper we present a broadcast-based coordination scheme that allows us to achieve multi-agent coordination without programming the behavior.

B. Swarm Coordination

Homogeneous systems of agents, by nature, lack a control structure for coordination. There exists no hierarchy of command by which lower ranked agents follow the actions of their superiors. For a many agent system, explicit message passing is likely to create a communication bottleneck. Only indirect communication, in the form of *broadcasts* or *cues*, offers a practical solution to the swarm coordination problem.

We define a *cue* as a prompt that a robot perceives from its environment. A *broadcast* consists of information that is transmitted indiscriminately, so that all robots receive the same information.¹ Using broadcast signals we show that we can create a "group dynamic" that all agents can sense and influence, but which does not reside in any individual.

As illustrated in Fig. 1, agents interact with their environment through sensors, actuators and broadcast signals. However, they are not permitted to address each other; additionally, the actions of an agent have no direct effect on other agents (or have explicit "interpretation").

The group dynamic is influenced and generated by the actions of all the robots, but is dominated by none in particular. It is sustained by the contribution of the population. It influences the behavior and decision of the robots, but allows the robots to function with local sensing, short broadcast communications, and no need for global information or maps. It has no power of direct actuation, but guides robots' behavior, like a group conscience.

The global dynamics of the system proposed will obviously have nonlinear characteristics. The system must be adaptive and sensitive to changes in individual "states" of the agents. To achieve such a system, we propose the use of signals we call "heartbeats," of which each agent has at least one, and which can respond to the collective behavior (or global dynamics of the environment/system) of all other robots' heartbeats.

¹For example, a locator beacon signal is a cue, while a robot's signal for help is a broadcast.

Different scenarios where explicit communications are combined with the "heartbeats" approach can also be considered to obtain more precise swarm coordination at the expense of cost and complexity. Agents may be given identities separate from their heartbeats so that it would be possible to ask one to turn its heartbeat off to elect itself a leader (see Section II-A). Or a group of agents could solicit another agent to join in by asking it to turn its heartbeat on.

Each agent broadcasts a unique *primary heartbeat* and responds to the collective behavior of all other heartbeats; they do not use the heartbeats to identify or address one another. We generate the ever-present, hardwired heartbeats with nonlinear oscillator circuits called van der Pol oscillators. The oscillators form a coupled network when each agent adds components of others' heartbeats to its own. Over a large range of frequencies and coupling factors, coupled van der Pol oscillators will synchronize their outputs [10]. This property is known as *frequency entrainment*.²

Bay and Hemami [3] showed that coupled van der Pol oscillators³ could be used to model the central pattern generators (CPG) that stimulate limb commands used in human walking and jumping. Cohen [6] used coupled oscillators to model the swimming motion of the sea lamprey. His research points to the plausibility that the swimming speed of a fish is controlled by an initial alteration of the frequency of an individual oscillator pair that results in all other oscillators entraining to the new frequency. We use van der Pol oscillators to mimic this type of behavior in our system of mobile robots. Other researchers have realized that CPG's, which control without central intelligence, hold tremendous potential for robotics applications. Crisman and Ayers [7] have formed a partnership to design a mobile robot suitable for operation in shallow water. The eight-legged walking machine is patterned after the American lobster, which is capable of walking in any direction, including laterally. Their CPG based controller coordinates and controls the robot's motion.

We exploit the frequency entrainment property to develop a global group dynamic that adaptively controls and coordinates a swarm of agents. Because synchronization is an inherent property of the coupled oscillators, we need not program the behavior. Synchronization occurs whenever agents "listen" to the heartbeats. While a more traditional approach such as broadcasting digital information over an ethernet is an option, it is a far more sophisticated solution than the coordination problem demands. Our coupled oscillator approach to the coordination of a swarm of agents is a more "natural" and simplistic solution.

This "reactive broadcast"-type of communication enables the swarm to

- sense when agents are added or dropped from the group;
- naturally choose a group leader;

²Frequency entrainment is a phenomenon that occurs when a periodic force is applied to a system whose oscillation is free and self excited, and the self-excited oscillation falls into synchronization with the driving frequency [3].

³Van der Pol oscillators are widely used for entrainment and biological modeling in past literature. This fact and characteristics such as robustness and simplicity motivated us to use VDP for our application.

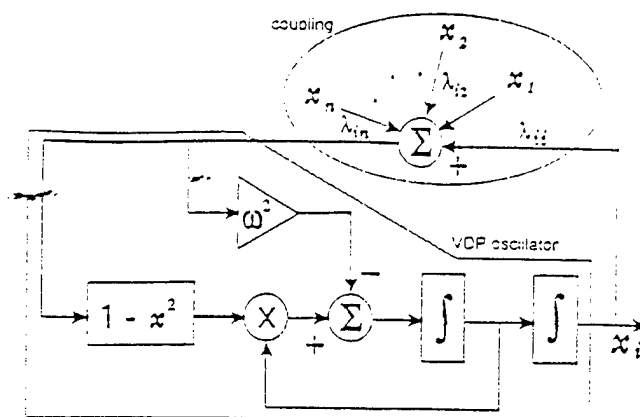


Fig. 2. Single van der Pol oscillator. (Coupling factors for a multiple oscillator scheme are also shown.)

- react to a single agent, i.e., any individual can effect a change in the group behavior.

C. Hardware Goals

We show that our broadcast-based coordination scheme can be used for multi-agent coordination in two ways. First, we demonstrate the synchronization of three agent heartbeats. We do this by realizing three van der Pol oscillators whose signals we broadcast and receive using an FM communication link. Our aim is to build an inexpensive hardware system whose performance closely matches simulated results. Next we use the coupled, three oscillator network to develop a global speed controller. We integrate one heartbeat circuit into an army ant, while allowing the other two circuits to operate as standalone (only one army ant was available for this experiment). We show that when the heartbeats couple, the agents are commanded to travel at a common speed. By increasing the coupling coefficients, we are able to effect a global increase in speed.

VDP oscillators will broadcast and receive heartbeats using an inexpensive FM communication system. The FM link introduces some distortion into the network of coupled oscillators, altering its behavior somewhat. However, the deviations produced by coupling the system with FM do not adversely affect performance goals of our system.

II. VAN DER POL OSCILLATORS

The nonlinear oscillators used in our distributed system are described by the well-known van der Pol equation which is used to describe an RLC circuit with a nonlinear resistor, or equivalently, a mass-spring-damper system with a position dependent damping coefficient [17]

$$\ddot{x} + \mu(p^2 - x^2)\dot{x} + \omega^2 x = 0.$$

The block diagram in Fig. 2 shows the construction of a single oscillator with $p = -\mu = 1$. It has two integrators, square-law and multiplier nonlinearities, and a gain parameter ω^2 that corresponds, roughly, to the squared frequency of oscillation. Coupling is effected through the summing junction.

Other signals x_j are coupled to oscillator i 's signal x_i through a coupling coefficient λ_{ij} , which can be positive

itory), or negative (inhibitory) (see also [5]). The absolute value of a coupling factor, λ , can range in value from 0 for no coupling to 1.0 for full coupling.

The primary broadcast heartbeat is invariant in order to avoid confusion when task-specific roles are assumed. The external signal(s) and any secondary broadcast signals may be modified by coupling terms to affect the agent's behavior. We may use as many different signal channels (each coupled with different network topologies) as necessary to accomplish our tasks, but we seek to keep these to an absolute minimum to minimize complexity and force the issue of emergent cooperative dynamics.

Use of van der Pol Oscillators in a Multi-Agent System

By adjusting the coupling coefficients between oscillators/agents, it is possible to use van der Pol oscillators for different purposes such as leader selection and synchronization.

1) *Leader Selection:* To implement our technique for leader selection, we suggest that the oscillator frequency given to each agent would be chosen randomly at the factory. If the frequencies vary continuously, then in all likelihood, each agent will have a unique frequency to differentiate itself, especially if the team size is relatively small (e.g., ≤ 10 in a population of 100 robots). Note that we still do not allow an agent to address another: the heartbeat frequencies are used for discrimination rather than identification.

2) *Synchronization:* Several phases (e.g., pallet lifting, entering or changing speed) of the Army Ant scenario require that the agents act as a team. Therefore, they must somehow synchronize their actions. The agents implicitly form a team when they coupled to the other's broadcast. As a team, they form a fully-connected network. The coupled network of oscillators can entrain so that the internal heartbeats of each agent oscillate at the same frequency as its teammates', regardless (almost) of the original frequency assigned to it.

The positively-coupled network is known to entrain to a common frequency for a broad range of individual frequencies [0], [15]. Fig. 3 shows a simulation of four waves of different intrinsic frequencies entraining with all coupling coefficients set to -0.3 . Then, one of the agents stops "listening" to its teammates, thus becomes the leader. The entrained frequency is different (higher) than any of the original frequencies. Note that entrainment requires the original frequencies to be reasonably similar to one another,⁴ although higher harmonic entrainment is possible [10].

Entrainment to a common frequency is used as the synchronization technique. When entrainment is detected, the common frequency may be used as a clock so that ensuing actions might be performed simultaneously thereafter. If we need to elect a leader that the others should follow (for example, when sectional heading must be arbitrated), then that leader can simply change its own coupling coefficients to zero, so that it oscillates at its original frequency, while the others remain entrained. This results in a set of waves that are entrained to

⁴Up to 30% deviation from the median frequency is permissible in order to obtain entrainment

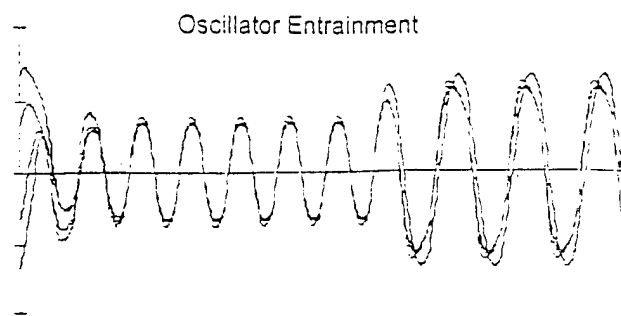


Fig. 3. Heartbeat entrainment to a common frequency: full coupling, and then, leader following. Note the change in amplitude due to decoupling.

the leader, as opposed to being mutually entrained; however, their phases may differ.

III. COORDINATION SCHEME

We demonstrate multi-agent coordination by realizing three heartbeat circuits and observing the entrainment behavior over a wide range of coupling factors. To achieve wireless coupling of the oscillators, we use an inexpensive FM communication link. While our results reflect some phase distortion in the received heartbeat signals, we show that the FM coupling does not adversely affect the performance goals of our system.

A. Oscillator Realization

The oscillator shown in Fig. 2 represents a heartbeat circuit, of which each agent has one. The input signals x_1 through x_n represent the heartbeats of all other agents whose signals are within "hearing" range. The received signals are coupled through a summing junction and added to the oscillator's feedback path.

For our research we use a three oscillator model, in which each heartbeat couples to two other heartbeat signals (Fig. 4). The heartbeat circuits are identical except that each is assigned a unique value for the frequency parameter ω^2 . For simplicity, we choose μ as unity. When μ is large it is extremely difficult to predict the regions of entrainment for a coupled oscillator network. It has been shown theoretically that for small μ ($\ll 1$) we can predict the regions of frequency entrainment for a van der Pol oscillator excited by a driving frequency. Simply stated, if the frequencies are not too different harmonic entrainment will occur. In the case of harmonic entrainment an oscillator synchronizes to the driving frequency. If the frequency separation is large, but the ratio of the frequencies is in the neighborhood of an integer or a fraction, frequency entrainment may still occur. The latter type of entrainment is called higher-harmonic or subharmonic entrainment since the oscillator entrains to a frequency that is a multiple or submultiple of the driving frequency [10]. When μ is made large the mathematical analysis of the van der Pol equation becomes very complex, and we are unable to predict the regions of entrainment. However, it has been shown through analog simulations that two mutually coupled van der Pol oscillators exhibit stable limit cycle oscillation for values of μ ranging from 0.1 to 1.0 [14]. Furthermore, our own simulations have indicated that frequency entrainment occurs for large

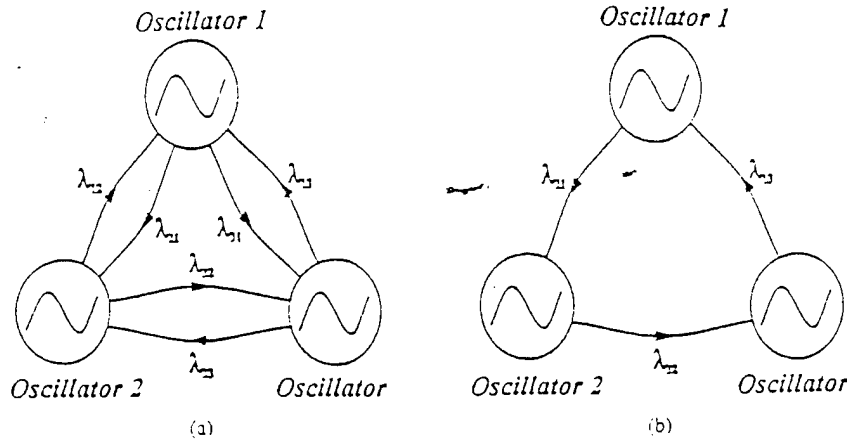


Fig. 4. (a) Full and (b) ring couplings of three oscillators.

numbers of mutually coupled van der Pol oscillators when $\mu = 1.0$.

While the oscillator realization is almost a straightforward adaptation of the block diagram, a direct synthesis of the block diagram in Fig. 2 yields a very low frequency oscillator with a limited frequency range. Thus, we time scaled all the oscillators by the same factor ($\times 1000$) and used ω^2 to tune each oscillator to a unique frequency.

B. Wireless Coupling

In choosing a communications medium to transmit agent heartbeats, several criteria were considered. The coordination scheme requires a communication system that is inexpensive, reliable, suitable for indoor or outdoor operation, and has reasonable range. There are several options for low cost wireless data links. For short range communication links, infrared and ultrasound are two popular media, but to synchronize a large swarm of agents spread out over a large area we need more range than these media offer. The primary disadvantage, however, in using infrared or ultrasonic communication links is that their directional transmission beams cannot provide reliable coupling. Object interference or an inability to maintain a line of sight are likely to prevent two agents from keeping their heartbeats locked. Thus, an omnidirectional communication link, such as an RF link, is most desirable for synchronizing a group of agents through mutual coupling.

In keeping with the Army Ant specifications, a simple and low cost RF communications link is used to couple the oscillators. Oscillator signals are broadcast using miniature tunable FM transmitters. Portable digitally tuned FM radios are used as receivers.

The range of our broadcast system is dependent upon the type of antenna used with the FM transmitters. The transmitters have a range of up to 1/3 mi with a 12-in wire antenna, but that range may be extended as much as 1 mi with a more sophisticated antenna. The system can be operated continuously for approximately two days on a 9 V battery.

C. Computer Control of Oscillator Parameters

To maintain autonomy in our Army Ant agents, we must give them the capability to control their own oscillator pa-

rameters. Some aspects of swarm coordination require more than just synchronization at a fixed frequency. The speed controller to be discussed in Section IV relies on changes in coupling strength to effect global changes in the speed at which agents are commanded to travel. Thus, agents shall be capable of changing their coupling strength, as well as coupling or decoupling themselves from the heartbeat and varying their frequency parameters. We assume that each agent can sense when to couple or decouple its oscillators. One obvious situation in which an agent would voluntarily decouple itself from the network is when it detects that its "health" is failing. For example, an agent's battery voltage can easily be monitored, so when the battery discharges to the point that it can no longer support a critical load such as the drive motors, the agent realizes its futility and takes appropriate action. Under normal circumstances, agents are not permitted to change their pre-assigned frequencies. Yet in the case that an agent detects a failure mode it can decrease its frequency to avoid any possibility of being elected a leader.

To provide automatic parameter control we use a voltage controlled amplifier interfaced to an up/down counter. Agents increase or decrease their parameters by either counter up or down. When an agent wants to decouple from the network it counts down completely, producing a control voltage of zero volts; consequently, incoming heartbeats are "coupled" with a gain of zero (i.e., $\lambda_{ij} = 0$ for $j \neq i$).

D. Experimental Results

We simulated three uncoupled oscillators with frequency parameters $\omega_1^2 = 1.2$, $\omega_2^2 = 1.3$, and $\omega_3^2 = 1.4$. Using the same settings, we acquired data from the hardware implemented oscillators and compared the results. Table I shows the operating frequencies for the simulated and actual oscillators. The operating frequencies of oscillator 1 and oscillator 2 matched the simulation within 2.5%. Oscillator 3 differed by 7.5% from the simulation. These errors could be made extremely low if precision resistors are used in building the oscillators, and gains are carefully trimmed to the exact values. Keeping in mind that a time scaling factor of 1000 was used, one sees that the frequency parameter *roughly* determines the oscillator frequency as expected.

TABLE I
OSCILLATOR FREQUENCIES FOR THE SIMULATED
AND ACTUAL THREE OSCILLATOR NETWORK

	Oscillator 1	Oscillator 2	Oscillator 3
Freq. Parameter	$\omega_1^2 = 1.2$	$\omega_2^2 = 1.3$	$\omega_3^2 = 1.4$
Simulated (Hz)	1245	1301	1352
Actual (Hz)	1272	1329	1454

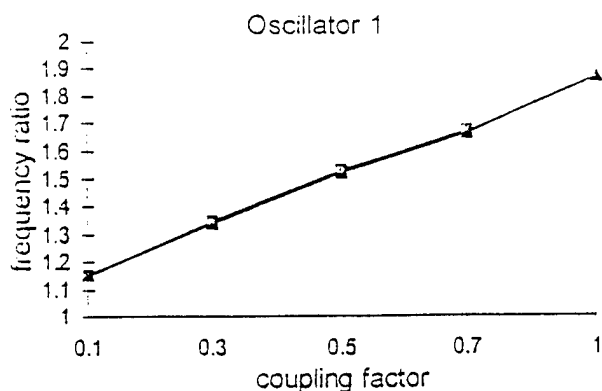


Fig. 5. Ratio of entrained frequency to natural frequency versus coupling factor for the simulated (▲) and actual (■) oscillator 1 ($\omega_1^2 = 1.2$).

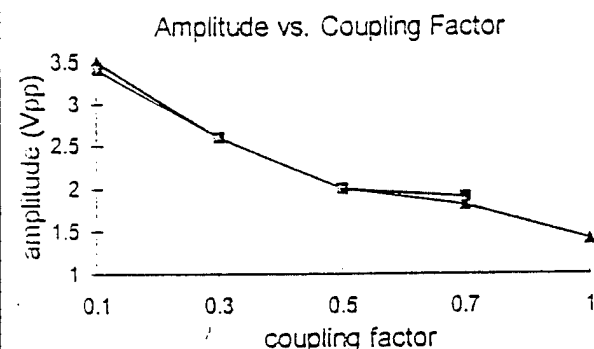


Fig. 6. Amplitude versus coupling factor for simulated (▲) and actual (■) oscillators.

We refer to the frequencies, ω_i , in Table I as the natural frequencies. In the following experiments we use the ratio of the entrained frequency to the natural frequency as a basis for comparing actual and simulated results. First we compare the results of our oscillator network, with coupling effected by hardwired connections, to the simulated network. We measured the entrained frequency and the signal amplitude for coupling factors ranging from $\lambda = 0.1$ to $\lambda = 1.0$. In each case all oscillators were set to the same coupling factor. In Fig. 5, which shows the entrained frequency versus coupling factor, we see that the performance of the actual oscillator matches the simulated results extremely well, with the frequency increasing slightly with increasing ω^2 . Comparable results were obtained for oscillators 2 and 3. The amplitude of the entrained frequency versus coupling factor is shown in Fig. 6. Two trends are evident from our data: the ratio of entrained frequency to natural frequency increases almost linearly as the coupling factor increases, while the amplitude decreases with increasing coupling strength.

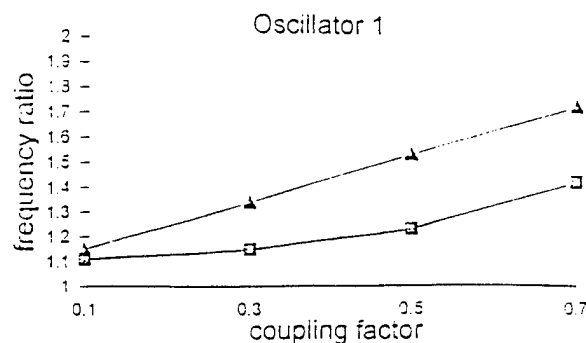


Fig. 7. Ratio of entrained frequency to natural frequency versus coupling factor for the simulated (▲) and actual (■) oscillator 1 using FM coupling ($\omega_1^2 = 1.2$).

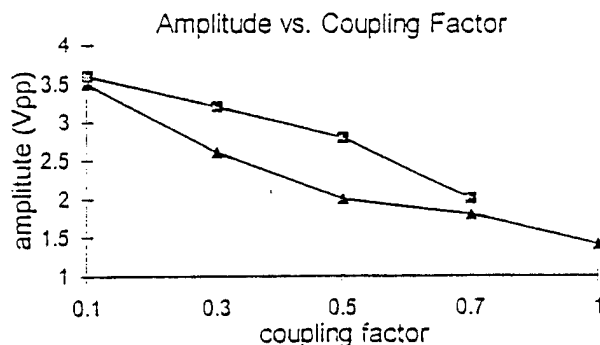


Fig. 8. Amplitude versus coupling factor for simulated (▲) and actual (■) oscillators with FM.

Figs. 7 and 8 show the results for oscillator 1 when the same experiments are repeated, but we use an FM link to couple the oscillators. Again, oscillators 2 and 3 behaved similarly to oscillator 1. With FM coupling the oscillators entrained to much lower frequencies than when a hardwired connection was used. Additionally, the increase in coupling strength did not have as much effect on the oscillator amplitudes in the FM coupled network. There is significant error between the performance of simulated oscillators and the FM coupled oscillators, but more importantly, the trends are still similar. Just as important is that the behavior of the FM system is consistent. The relationship between frequency ratio and coupling factor is slightly more linear for the simulated oscillators. The frequency ratio versus coupling factor data for the FM coupled oscillators looks similar to the simulated results, but rises with a smaller slope. The error is greatest at the highest coupling factor. The amplitude data starts out with simulated and FM coupled oscillators matching very well at low coupling, has the greatest error between $\lambda = 0.3$ and $\lambda = 0.5$, and shows little error at higher coupling factors.

A large portion of the error in the FM data can be attributed to the manner in which the data was acquired, rather than to the FM system. When the heartbeat circuits were connected to the data acquisition board, the transmitters were as close as 1 foot from each other. Unlike most miniature FM transmitters, which have a 5 mW output power, our transmitters have a 75 mW power output. While the higher output power affords a greater broadcast range, at close range there is considerable interference between devices. Distortion is noticeable in the

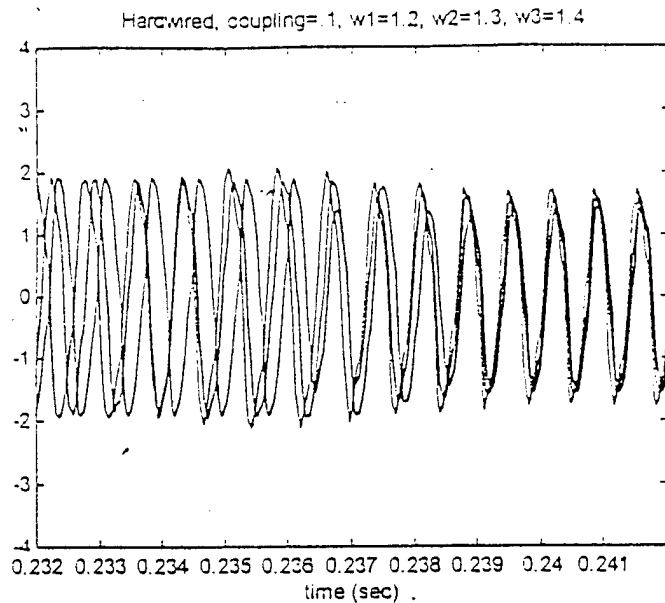


Fig. 9. Hardwired coupled oscillator network entrained to 1472 Hz ($\lambda = 0.1$). Oscillators begin to entrain at 0.236 s.

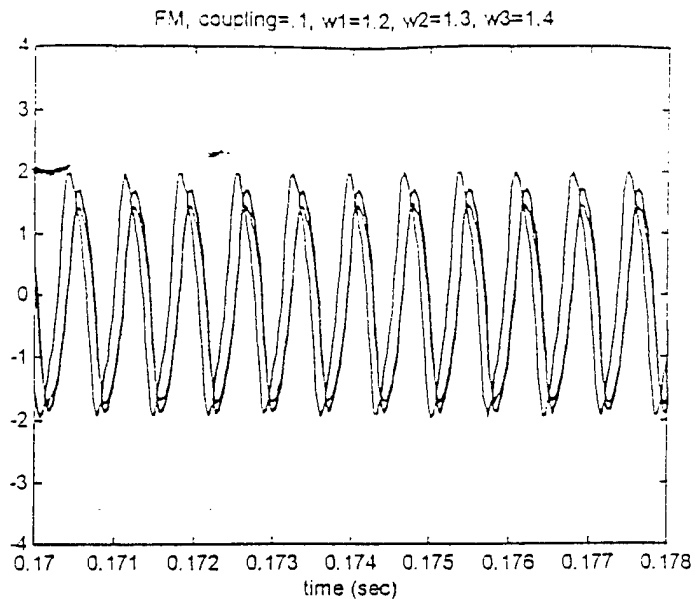


Fig. 10. FM coupled oscillator network entrained to 1415 Hz ($\lambda = 0.1$).

received heartbeat signals, so we expect the behavior of the van der Pol oscillators to be affected. In a scenario where agents are randomly distributed rather than clustered together, there is less interference and the behavior of the FM coupled system is likely to improve.

Also, the presence of the radio link can be modeled as a complex system block in the oscillator diagram of Fig. 4. If we neglect this block, we are neglecting a significant portion of the overall dynamics and yet we still expect the oscillators to entrain as theoretically predicted.

While amplitude distortion is a problem in AM systems, FM systems are plagued by phase distortion. The phase distortion results when the phase relationship between the carrier and the sidebands are altered [11]. Phase equalization networks can be used to correct this problem. Of course a price is paid for higher quality signal processing. One disadvantage of using inexpensive off-the-shelf communications hardware is that circuit schematics and specifications are often unavailable, and it is difficult to know exactly what type of performance to expect. Nevertheless, our broadcast-based coordination scheme does not demand a high quality FM link. We are more concerned that the oscillators stay entrained and that the network behaves accordingly to changes in coupling factor, than we are with the specific entrainment frequency.

Figs. 9 and 10 show a three oscillator system entrained to a common frequency for a coupling factor of 0.1 using both hardwired and FM coupling. In the hardwired case we captured the data as the oscillators moved from free oscillation to full entrainment. We see that there is a brief transient that occurs between the time when the oscillators first couple until they settle to the entrained frequency. The transient was too long in the FM case to display the transition in one frame. Thus, the graph shows the oscillators after they are fully entrained. Fig. 11 shows the behavior of the network in a leader-follower configuration. We elect oscillator 2 as leader and see that the leader does not change its behavior, but

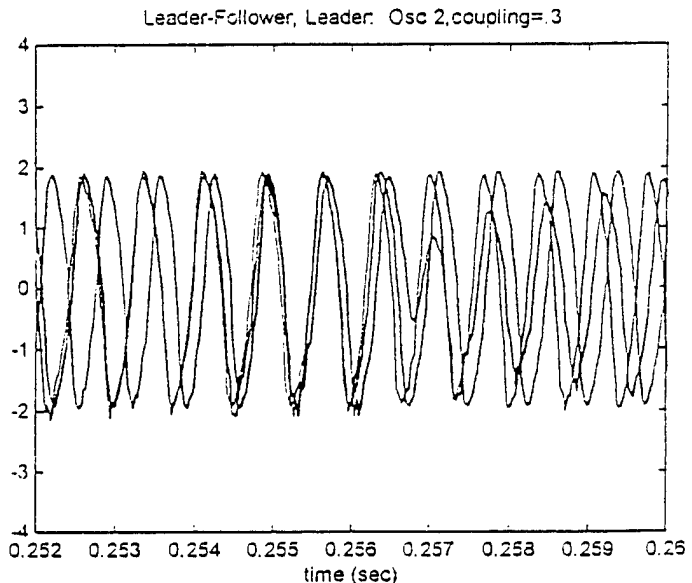


Fig. 11. Leader-Follower configuration with oscillator 2 (dashed line) as leader ($\lambda = 0.3$). Oscillators entrain in frequency but not in phase.

the followers entrain to the leader's frequency. Unlike in a mutually coupled configuration, when oscillators entrain to a leader their entrained frequencies are not in phase.

IV. GLOBAL SPEED CONTROL

Our broadcast-based coordination scheme will be applied, in many cases, to simultaneously initiate a common action within a swarm of Army Ants. Used in this manner, the entrained frequency serves only as a trigger; as long as all the agents' heartbeats synchronize, their coupling strength is irrelevant. The global speed controller requires two properties of mutually coupled van der Pol oscillators. To control agents' speed we require synchronization behavior so that all agents travel at the same velocity. Also, we employ a property presented in

our results from Section III: the entrained frequency increases almost linearly with increasing coupling factor. Thus, we effect a global change in speed by changing the strength at which heartbeats couple.

The global speed controller application was chosen to demonstrate that our method could be used to trigger other behaviors. By detecting thresholds in dynamically entrained variables, practically any behavior can be symbolically triggered. Similarly, the same approach can be used to control many behaviors or decisions. This also shows how a local (individual) decision can affect a global change.

In this section, we explain the importance of global speed control to the Army Ant scenario. Then, using the three heartbeat circuits we show how to implement the global speed control. We discuss the additional hardware required to interface the heartbeat circuit to motor controller chips and to the Army Ant's processor, the 68HC11 microcontroller [2]. Lastly, with one heartbeat integrated into an army ant we use the global dynamic created by the three heartbeats to control the robot's speed. The performance of the speed controller is evaluated for several different coupling factors.

One situation where it is desirable to have all agents moving at the same velocity is when agents must cooperatively carry a pallet. Assuming that agents beneath a pallet can align themselves along the direction of motion (see [18]), we would like them to travel at a common speed while carrying the load. A pallet only rests on top of a group of agents: it is not rigidly attached. In a group of agents that are not traveling at uniform speed, slower moving agents may eventually lose contact with the pallet as faster agents carry it away.

While not as critical as the pallet carrying example, global speed control is useful for some ground maneuvers. Suppose a group of agents must search a large area. Agents traveling at the same speed will cover roughly the same amount of ground and expend their resources at the same rate. The intention of the Army Ant swarm is to accomplish goals cooperatively, with no one agent assuming a greater role than another. By exercising speed control we can better maintain homogeneity in the level of activity of the swarm.

As we have previously discussed, an important characteristic of our coordination scheme is that it reacts globally to local changes. If for some reason one agent in the heartbeat network senses that it must decrease its speed it will lower its coupling strength, causing the entire heartbeat dynamic to oscillate at a lower frequency. The global reaction of the network implies that if a group of agents is carrying a pallet toward an obstacle, only a few agents in front need to detect the danger in order to avert disaster. As soon as the few agents begin to slow, their reaction will be sensed by the heartbeat dynamic and every member of the group will decrease its speed accordingly.

A. Hardware Implementation

To realize our global speed controller we need a way to transfer information from the heartbeats to the motor controllers. We show that this is accomplished very easily using a frequency-to-voltage converter. Also, we want the speed to be affected only when the heartbeats are coupled. When

heartbeats are oscillating freely agents are not considered to be part of a collective and do not need to be coordinated.

1) Heartbeat to Motor Controller Interface: The first stage of our interface is a frequency-to-voltage (F/V) circuit. A frequency-to-voltage converter outputs an analog voltage that is proportional to the frequency of the input signal. The gain, specified in volts per Hz, is user defined by a few external resistors and capacitors. Our motor controller chips require a digital velocity command. Therefore, we need to insert an analog-to-digital converter between the output of the F/V converter and the motor controller. The motor controller outputs a pulse width modulated signal to an H-bridge circuit, which drives the motors. The 68HC11 microcontroller includes an analog-to-digital converter; so with only one additional integrated circuit (the F/V converter) we can interface a heartbeat circuit to an army ant's motor control hardware. The interface provides a velocity command to the motor controller that varies linearly with the heartbeat frequency of oscillation.

The question arises: How do we keep the frequency of uncoupled oscillators from influencing agents' speed? The key lies in the fact that, in a mutually coupled system of van der Pol oscillators, the entrained frequency will always be greater than any of the original free frequencies regardless of the coupling factor. Consequently, we can set a frequency "trip point" that is slightly greater than any free frequency in a network. Assuming that the frequency separation between any two robot heartbeats is small, the trip point can be fixed. Our motor control algorithm issues a "do nothing" command when voltages at or below the voltage corresponding to the frequency trip point are received from the F/V converter. An output from the F/V converter that is above the trip point indicates that the system has coupled. Only then does the velocity command to the motor change proportionally with the heartbeat frequency of oscillation.

Each army ant shall have the described interface between its heartbeat circuit and its motor control electronics in an actual scenario. In this experiment we use three mutually coupled heartbeats with only one heartbeat integrated to an army ant. Thus, we acquire velocity data on only one agent. We have already shown that the heartbeats will entrain to a common frequency. Given that the trip points and frequency-to-voltage gains are the same in every interface, we can assume that the global heartbeat dynamic effects the same speed command in all agents.

B. Experimental Results

The control of the Army Ant drive motors for this experiment is open loop with respect to velocity, i.e., we do not utilize tachometers to provide feedback. Therefore, only commanded signals are at our disposal. We choose to acquire the motor velocity command data to evaluate our global speed controller rather than the signals to the motors. The motor drive signals are pulse width modulated, and not as convenient a format. In the first set of experiments we use hardwired coupling, and observe the commanded speed to the agent's drive motors as its heartbeat transitions from freely oscillating to fully entrained. We repeat the experiment for

TABLE II
PREDICTED AND ACTUAL SPEED DATA FOR HARDWIRED COUPLED SYSTEM

coupling factor (λ)	entrained frequency (Hz)	predicted speed (rpm)	actual speed (rpm)
0.3	1724	10.50	10.45
0.5	1950	11.38	11.32
0.7	2136	13.00	12.99

three different coupling factors: $\lambda = 0.3$, $\lambda = 0.5$, and $\lambda = 0.7$. As expected, the speed command increases with increasing coupling strength.

Given the entrained frequency data collected in the last section, and the conversion gains used in the speed controller, we can predict the speed commands for each coupling factor.⁵ Note that predicting the speed at which coupled agents will travel is not a goal of the global speed control application. The object of the speed controller is to ensure that all agents move at a common speed, and that any change in an individual's speed will be reflected in every agent whose heartbeat is part of the dynamic. The entrained frequency is dependent upon the number of agents in the network. For large systems it may be inconvenient to predict the speed, and impossible if the number of agents is unknown. Besides, the Army Ant concept does not support deterministic, centrally computed knowledge. Yet comparing the predicted speed to the data we acquired provides a useful check to validate our controller.

Table II shows the predicted speeds for the three coupling factors based upon the entrained frequencies acquired in Section III, as well as the speed data collected in these experiments. The fact that the predicted data matches the actual data so closely indicates that the frequency-to-voltage conversion remains linear over a wide frequency range.

The same experiments were conducted using FM coupling. While we can expect the entrained frequency to remain constant for a given coupling factor and a set number of agents when hardwired coupling is used, this is not the case when heartbeats are coupled through FM. Just as we experience degrees in the quality of reception on our FM radios, the performance of our broadcast-based system is subject to vary somewhat with external interference. Therefore, it is not very meaningful to compare the predicted speed to the actual speed in the FM case. We have already validated the speed controller with our hardwired data, so we present the FM data in a slightly different manner. Using the speed data acquired with FM coupling, we can work "backward" and estimate the entrained frequencies. The entrained frequencies from these experiments, when compared to the data from the hardwired coupled system, provide an example of how the performance of the FM system can deviate. Table III summarizes the data for three different coupling factors.

The error between the acquired speeds for the hardwired and FM cases is under 5% for $\lambda = 0.3$ and $\lambda = 0.5$ and under 10% for $\lambda = 0.7$. The A/D output, which provides the speed command to the motor controller chip, was acquired and

⁵ The entrained frequency divided by the F/V gain gives an analog voltage. This voltage, when multiplied by the maximum rpm (25 rpm) and divided by the maximum A/D voltage (5 V), provides the expected speed.

TABLE III
SPEED DATA FROM FM COUPLED SYSTEM

coupling factor λ	entrained frequency <i>hardwired</i>	entrained frequency <i>FM</i>	speed <i>FM</i>
0.3	1507	1674	10.15
0.5	1590	1844	11.23
0.7	1823	1964	11.91

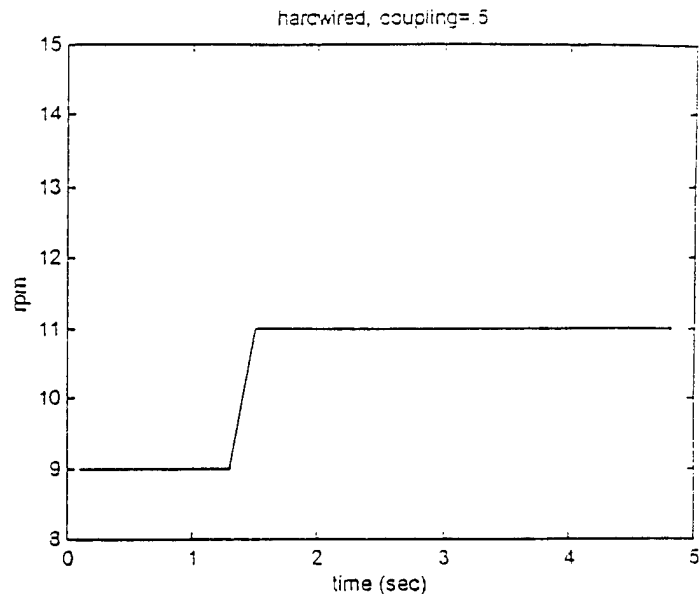
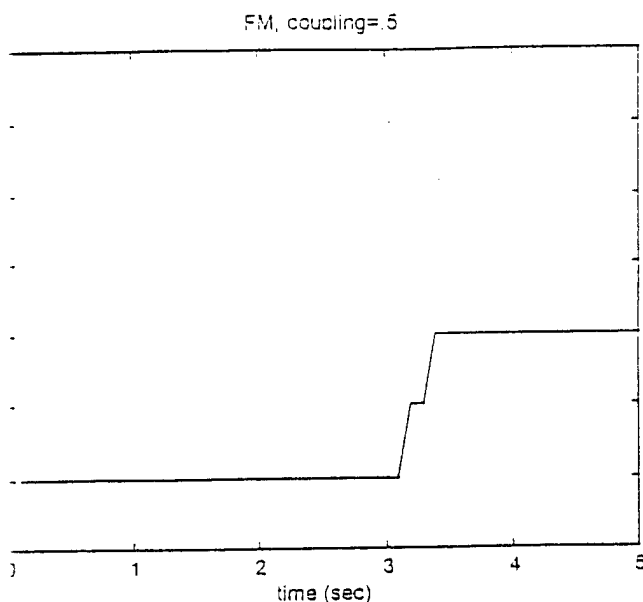


Fig. 12. Global speed for a hardwired oscillator network with a coupling strength of $\lambda = 0.5$.

converted to an rpm value in our software routine. Figs. 12 and 13 show plots of the commanded speed for cases in which the heartbeats were coupled via hardwired connections and FM. At an arbitrary time during the data acquisition the oscillator receivers were turned on. The subsequent step in the speed occurs as the system transitions from an uncoupled to a coupled system. The level of the commanded speed of the coupled system is dependent upon the coupling factor used. Notice that the graphed speeds are all integer values. All velocity commands were truncated by the algorithm during implementation. The velocity data shown in Tables II and III was obtained through examination of the raw output from the A/D.

We have shown that the heartbeat dynamic can easily be applied as a global speed controller. To interface the heartbeat circuit to an agent's motor control hardware, only one additional IC is required. The essence of the speed controller interface, the frequency-to-voltage converter, exhibited a highly linear response over a wide range of input frequencies.

The performance of the speed controller compared favorably to predicted results. We showed that changes in coupling strength produce the same directional trend in the commanded velocity to an agent's drive motors. Additionally, we have verified that the error introduced by coupling the system through an FM communication link is at an acceptable level. In short, our broadcast-based coordination scheme was successfully and simply applied to one of the more complex



3. Global speed for an FM coupled oscillator network with a coupling gain of $\lambda = 0.5$.

vivors requiring coordination. The results attained in our control experiments serve to reinforce our confidence the heartbeat concept is a viable method for coordinating actions within the Army Ant swarm.

V. OTHER APPLICATIONS/USES

The applications of coupled nonlinear oscillators in the Army Ant scenario are not limited to leader selection and synchronization. In this section we explain the use of similar oscillator schemes for error detection.

Error Detection in Distributed Load Bearing

To lift and carry a palletized load, Army Ant robots need to work together to establish a stable transportation mechanism. The method for controlling the collective behavior of Army Ants for load transportation is described in [18]. It is achieved by selecting a leader, without direct communications, using force sensors mounted at the point of contact with the load. The method described in [18] results in stable behavior of the horizontal movement of agents under the pallet. The method described here differs in the type of communication: in [18], coupling is a physical force transmitted by the load; here, we use a broadcast signal which can be used without physical contact at all.

During lifting and transportation of pallets it may be important that the pallet be supported equally (to some degree) by all robots. It is possible to detect this condition by comparing the sensed vertical force on agents. However, since there is no central controller, this has to be done collectively. Assuming the robots under the pallet can receive signals from their teammates, they can form a ring or fully coupled network where the feedback gains of the oscillator are linear functions of the output of vertical force sensors (Fig. 4).

The addition of a variable feedback gain to the oscillator network enables the robots to detect whether the pallet is level

and/or the load distribution is even. Feedback gain factor λ_{ii} changes over time as a function of the output of the force sensor. Therefore, the feedback gain of the each oscillator changes according to the vertical force component sensed by the robot.

Using the incoming signals from other robots and force sensor output, robots can determine the difference e_i between their own sensor and the sensor of the "previous" robot (or the average of all incoming signal in the full coupling case). In other words, the signal e_i is a relative measure of discrepancy in force sensor outputs. Robots that detect a difference beyond the predefined parameters can broadcast a warning signal. Transportation starts and continues as long as there is no robot broadcasting a warning signal caused by this or any other problem.⁶

$$\text{The error signal: } e_i = \lambda_{ii}x_i - \sum_{j=i} \lambda_{ij}x_j$$

(In ring coupling, $j = i - 1$ only).

In the following, the oscillators use ring coupling and all coupling coefficients are set to -0.3 . The feedback gains λ_{ij} vary according to the linear mapping⁷ from internal sensor to gain. When all gains are approximately equal, entrainment is reached after a time with frequencies in the range 0.8–1.2. For our purpose, we set all oscillator frequencies ω^2 to 1. However, the amplitude of the oscillations will differ from agent to agent if the feedback gains are not the same. Each agent is able to compare its output with the output signal of the coupled teammate. In addition, the ring configuration enables each robot to detect the sensor output variations occurring in any other teammate by comparing its sensor output only to the previous robot in the configuration. Because all such comparisons accumulate as coupling signals pass along the network, the errors in feedback gains (force sensors) can be detected by other oscillators, even if they are not nearby. The robots may adjust their oscillator feedback gains by exerting more or less force or by moving to more desirable spots—thereby equalizing vertical forces until oscillators agree.

In full coupling, the entrainment is much faster than in the previous case because of the multiple effects on each agent. The coupling coefficient is chosen smaller (~ 0.1) to reduce the strength of incoming signals. The error signal is much smaller in magnitude for a fully coupled network. For this reason, in error detection applications, a ring coupled network may be preferred. The difference between full and ring coupling methods are summarized in Table IV.

It may be necessary to adjust the allowable ranges for error signals so that problems may be detected regardless of the number of agents. As long as the number of agents is greater than four, the same parameters for allowable ranges are likely to work for all cases. When additional oscillators are successively added to the system, the change in the signal levels is less and less significant. Addition of new oscillators to a ring network of eight oscillators does not have a perceivable

⁶Of course, an explicit communication method could be used to detect uneven load distribution at the expense of added complexity and cost.

⁷Maximum and minimum values of the sensor output corresponds to gains 1.3 and 0.7, respectively.

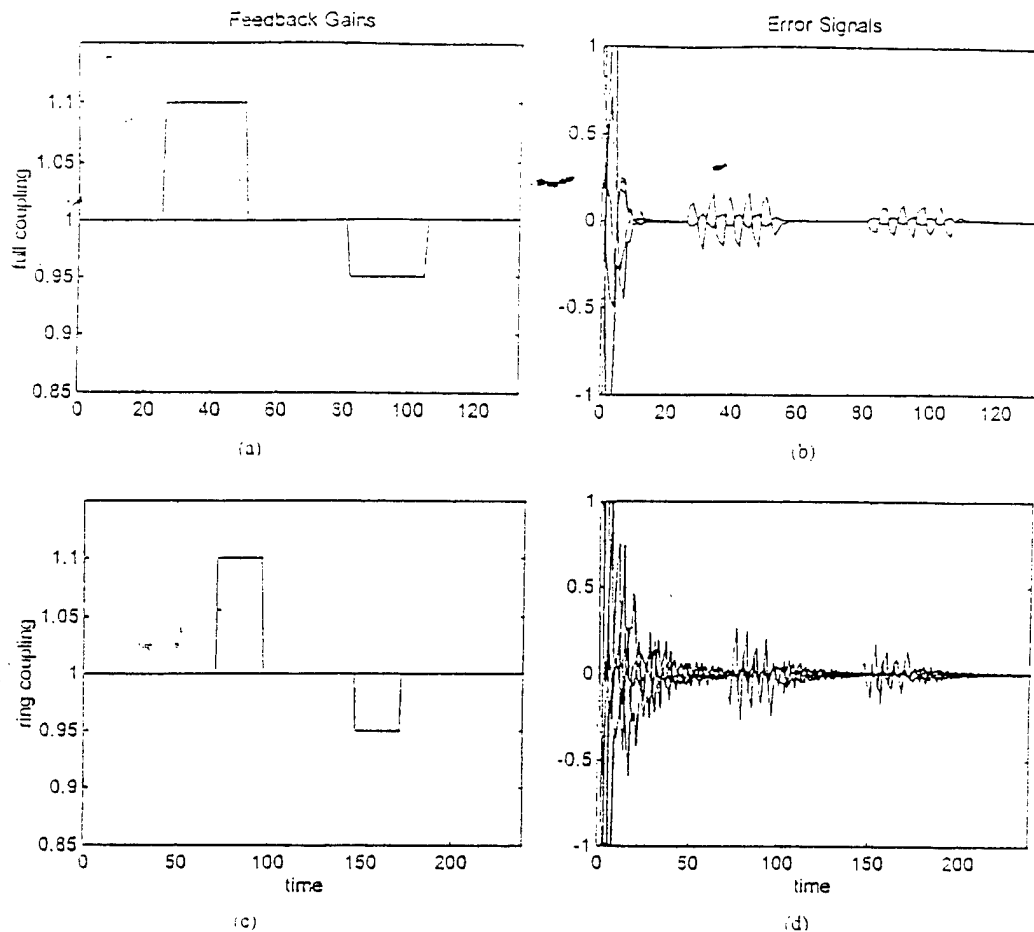


Fig. 14. Feedback gains and error signals of five oscillators in full (top) and ring (bottom) coupling ($\lambda = 0.1$ for full, $\lambda = 0.3$ for ring configuration).

TABLE IV
DIFFERENCES BETWEEN FULL AND RING COUPLING METHODS

	Full Coupling	Ring Coupling
communication type	broadcast	filtered using ID numbers
error signal: level	relatively low	relatively high
propagation delay	no	yes
coupling coeffs	~ 0.1 or less	~ 0.3
entrainment	fast	slow
additional requirements	total number of robots in a team must be known for averaging purposes	agents must "choose" another for coupling

effect. Similar characteristics are observed in the period and amplitude of the entrained oscillators. Furthermore, a discrete mapping function from sensor output to the feedback gain may improve the detection of range violations.

In simulation examples given, we changed the feedback gain of only one agent, and plotted e_i 's computed by all agents. As shown in Fig. 14, at least two agents are able to detect the error in both coupling methods, although detection by one agent is sufficient for the swarm to become aware of the problem: a large value of the magnitude of error signal (in one

or more agents) is an indication of uneven load distribution. The threshold value for error detection is not the same for both coupling configurations. Because the error signal is much smaller in magnitude for a fully coupled network, the threshold value must be chosen accordingly.

The stability of the coupled oscillator system given here is a theoretically unproved issue. We are aware of no such proof for our general oscillator circuitry. With the addition of unmodeled RF circuit dynamics, such a proof is unlikely.

VI. CONCLUSION

Distributed multi-agent systems such as the army ants offer new and promising possibilities in the application of robotics to industry. Despite the many advantages inherent to distributed robotics systems, they have traditionally been very difficult to coordinate and control due to the absence of a central supervisor or a hierarchy of command. Agents in a distributed system must be capable of collectively accomplishing tasks using only locally sensed information and little or no direct communication. Toward this goal, this paper has introduced a broadcast-based coordination scheme that provides a global group dynamic that can control individual agents, is influenced by all agents, but does not reside in any agent.

Our scheme addresses many of the coordination problems that exist in the Army Ant scenario without compromising

of the project's principles. In adhering to the principles of homogeneity and distributed control we maintain all of the benefits associated with the Army Ant concept.

In the fully-coupled broadcast-based coordination scheme, agents are never addressed directly; all communication is indirect, in the form of broadcasts. Unlike direct communication systems which require a larger bandwidth as the number of agents increases, our method is completely scalable. The group dynamic automatically adapts to changes in the number of agents in a coupled network. In fact, it reacts only to the composite of all the coupled heartbeat signals without regard to the number of heartbeats. As such, the scheme is ideally suited for a dynamic swarm, wherein army ants adjust their size to fit the task at hand.

While our coordination scheme requires the assignment of a unique heartbeat frequency to each agent, we in no way violate agents' homogeneity. The heartbeats are never used for the purpose of identification. They can, however, be used for discrimination during leader selection. We have shown that our scheme can be used to entrain agents to a leader's frequency. Yet, most of the proposed applications of the coordination scheme (e.g., synchronized steering, lifting, and speed control) only require entrainment to any common frequency. Furthermore, each heartbeat can replace any other heartbeat in a coupled network, and the failure of a heartbeat circuit results only in that particular agent being excluded from the coordination effort. Thus, our coordination method preserves the innate robustness of the Army Ant swarm.

Just as the Army Ant swarm is a reactive/behavior-based system, so too is the coordination scheme's network of coupled heartbeats. The proposed method does not require any reasoning or planning. Rather, we exploit the known properties of coupled nonlinear oscillators to create the global group dynamic that controls agents' actions. The dynamic naturally adapts to any changes in the network: because the dynamic consists of the composite signal of all heartbeats in a network, it is sensitive to variations in individual heartbeats. The reactive nature of our scheme allows the army ants to respond quickly in a dynamic environment.

The primary benefit of this research is that the broadcast-based coordination scheme was validated by the actual construction and testing of a system of heartbeat circuits. By realizing the concept with actual hardware, we are forced to avoid making assumptions that may later prove impractical. Some of the results are summarized as follows:

- The behavior of the realized heartbeats compared favorably to the predicated behavior that was obtained through simulation. An actual network of three heartbeats was shown to entrain either to a leader frequency or to a common frequency over a wide range of coupling strengths, in the presence of completely unmodeled transceivers.
- It is possible to broadcast and receive heartbeats using an inexpensive FM communication system. The FM link introduces some distortion into the network of coupled oscillators, altering its behavior somewhat. The deviations produced by coupling the system with FM do not adversely affect performance goals of the system.

- The broadcast-based coordination scheme can be used to synchronize many actions such as lifting or steering. It can also be applied to more complex behavior, as demonstrated in the global speed controller example. We have demonstrated a simple controller that detects when an agent is part of a global group dynamic then allows the dynamic to control an agent's speed. In this manner, all agents in a network travel at the same speed and increase or decrease their speed in accord.
- A proposed technique for error detection was devised which would allow one robot to adapt its contribution to better match that of its teammates. This automatic error detection and correction acts as a regulating group dynamic.

There are a few areas relating to the coordination of the Army Ant swarm that require further research:

- Further investigation on the conditions under which agents will be permitted to vary their heartbeat parameters. Algorithms for executing this logic must be developed and tested.
- The process of electing a leader deserves further consideration. Although we want to avoid a fixed hierarchy, some form of hierarchy in form of "temporary leaders" results. Additionally, we must investigate how to implement the idea of selecting as leader the agent with the highest heartbeat frequency with a minimum of communication and complexity.
- The use of AI methods for adaptive coupling parameters may prove useful. To create more robust and "intelligent" system, genetic algorithms and classifier systems may be used to realize a method for on-line computation of necessary coupling coefficients for a particular task and/or situation.

REFERENCES

- [1] T. Balch and R. Arkin, "Communication in reactive multiagent robotic system," *Autonomous Robots*, vol. 1, pp. 27-32, 1994.
- [2] J. S. Bay, "Design of the 'army-ant' cooperative lifting robot," *IEEE Robot. Automat. Mag.*, vol. 2, no. 1, Mar. 1995.
- [3] J. S. Bay and H. Hemami, "Modeling of a neural pattern generator with coupled nonlinear oscillators," *IEEE Trans. Biomed. Eng.*, vol. BME-34, no. 4, pp. 297-306, 1987.
- [4] G. Beni and J. Wang, "Theoretical problems for the realization of distributed robotic systems," in *Proc. IEEE Int. Conf. Robotics Automation*, Apr. 1991, pp. 1914-1920.
- [5] H. L. Chiel *et al.*, "Robustness of a distributed neural network controller for locomotion in a hexapod robot," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 293-303, June 1992.
- [6] A. Cohen, "The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model," *J. Math. Biology*, vol. 13, pp. 345-369, 1982.
- [7] J. Crisman and J. Ayers, "Implementation studies of a biologically-based controller for a shallow water walking machine," in *Proc. SPIE*, 1992, vol. 1831, pp. 212-222.
- [8] J. L. Deneubourg and S. Goss, "Collective patterns and decision making," *Ethology, Ecology, Evolution I*, pp. 295-311, 1989.
- [9] N. R. Franks, "Army ants: A collective intelligence," *Amer. Scientist*, vol. 77, pp. 139-145, Mar. 1989.
- [10] C. Hayashi, *Nonlinear Oscillations in Physical Systems*. New York: McGraw-Hill, 1964.
- [11] H. Krauss, C. Bostian, and F. Raab, *Solid State Radio Engineering*. New York: Wiley, 1980.
- [12] C. R. Kube and H. Zang, "Collective robotics: From social insects to robots," in *Proc. IEEE/RSJ Int. Conf. Intel. Robots Systems*, 1993.

- [13] P. N. Kugler, and M. T. Turvey, *Information, Natural Law, and the Self-Assembly of Rhythmic Movement*. Hillsdale, NJ: Lawrence Erlbaum, 1987.
- [14] D. A. Linkens, "Analytical solution of large numbers of mutually coupled nearly sinusoidal oscillators," *IEEE Trans. Circuits Syst.*, vol. CAS-21, pp. 294-300, 1974.
- [15] ———, "Stability of entrainment conditions for a particular form of mutually coupled van der Pol oscillators," *IEEE Trans. Circuits Syst.*, vol. CAS-23, Feb. 1976.
- [16] M. Mataric, "Distributed approaches to behavior control," *SPIE-Sensor Fusion V*, vol. 1823, pp. 373-381, 1992.
- [17] J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
- [18] D. Stillwell, and J. S. Bay, "Toward the development of a material transport system using swarms of ant-like robots," in *Proc. IEEE Int. Conf. Robotics Automation*, 1993, pp. 766-771.
- [19] C. Ünsal and J. S. Bay, "Spatial self-organization in large populations of mobile robots," *Proc. 9th IEEE Int. Symp. Int. Circ.*, 1994, pp. 249-254.



Cem Ünsal (S'92-M'93) was born in Ankara, Turkey, in 1967. He graduated from Lycée de Galatasaray, Istanbul, Turkey, in 1986. He received the B.S. degree with honors in electrical and electronics engineering from Bogaziçi University, Istanbul, in 1991, and the M.S. and Ph.D. degrees in electrical engineering from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, in 1993 and 1997, respectively.

While at Virginia Tech, he was a Research Assistant with the Center for Transportation Research. He is currently a Postdoctoral Fellow with the AHS Group, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. His research interests include multirobot systems, nonlinear systems control, automated highway systems, intelligent vehicle control, and learning automata.

Dr. Ünsal is a member of AMS and ITS America.



Kimberly Sharman Evans received the B.S.E.E. and M.S. degrees from Virginia Polytechnic Institute and State University, Blacksburg, in 1990 and 1994, respectively.

From 1990 to 1992, she was with the Navy Center for Space Technology at the Naval Research Laboratory (NRL), working in the area of electronics design for spacecraft power systems. From 1994 to 1996, she was Director of the Multimedia Division, Omniview Inc., Knoxville, TN, working on applications for the company's 360° imaging

technology. She is currently with Digital Directory, Knoxville



John S. Bay (S'83-M'84-SM'97) received the B.S.E.E. degree from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, and the M.S. and Ph.D. degrees in electrical engineering from The Ohio State University, Columbus.

He is currently an Associate Professor of Electrical Engineering at Virginia Tech, where he has taught and performed research in control systems and robotics since 1989, and is director of the Robotics and Machine Intelligence Laboratories.

His past research interests include kinematics and dynamics, nonlinear dynamical systems and control, distributed artificial intelligence and sensor-based robotics. He is currently working in the area of intelligent, many-agent dynamics, machine learning, and multirobot coordination and control. He serves as Associate Editor for *IEEE Control Systems Magazine*.

Dr. Bay is a member of the ASEE.

CALVIN: Winner of the Fourth Annual Unmanned Ground Vehicle Design Competition

Matthew Caprio
University of Texas

Susan Larkin
Lucent Technologies

John Bay, Paul Johnson, Scott Wenger,
Christopher Johnson, and Charles Reinholtz
Virginia Tech

ABSTRACT

The Unmanned Ground Vehicle Competition is jointly sponsored by the SAE, the Association for Unmanned Vehicle Systems (AUVS), and Oakland University. College teams, composed of both undergraduate and graduate students, build autonomous vehicles that compete by navigating a 139 meter outdoor obstacle course. The course, which includes a sand pit and a ramp, is defined by painted continuous or dashed boundary lines on grass and pavement. The obstacles are arbitrarily placed, multi-colored plastic-wrapped hay bales. The vehicles must be between 0.9 and 2.7 meters long and less than 1.5 meters wide. They must be either electric-motor or combustion-engine driven and must carry a 9 kilogram payload. All computational power, sensing and control equipment must be carried on board the vehicle. The technologies employed are applicable in Intelligent Transportation Systems (ITS).

A written design report and an oral presentation are required from each team, and expert judges evaluate these along with inspecting the actual vehicles. Design judging focuses primarily on the design process rather than the implementation of that design in the actual vehicle. The latter feature is evaluated by performance on the obstacle course. The team winning the design contest receives a \$1000 award from SAE and is offered the opportunity to present their design paper at the SAE World Congress. The 1996 competition was held at Walt Disney World in Orlando on July 13-15.

This paper presents the conceptual design of the vehicle and its components. Innovative aspects of the design are highlighted, along with descriptions of the electronics, software, computers, actuators, sensors, and the means of system integration. The steps followed in the design process are described along with the use of computer-aided design. Considerations of safety, reliability, and durability are included. The analyses leading to the predicted performance of the vehicle (speed, ramp climbing, reaction times, etc.) are also documented. Although not a factor in judging, the paper also includes a cost estimate (not counting student labor) for the final product if it were to be duplicated.

INTRODUCTION

CALVIN (Computerized Autonomous Land Vehicle with Intelligent Navigation), shown in Figure 1, is a battery powered

three-wheeled vehicle with a computer vision system for line following and ultrasonic sensors for obstacle avoidance. CALVIN was one of two vehicles entered by the Virginia Tech Team in the 4th Annual International Autonomous Ground Vehicle Competition.

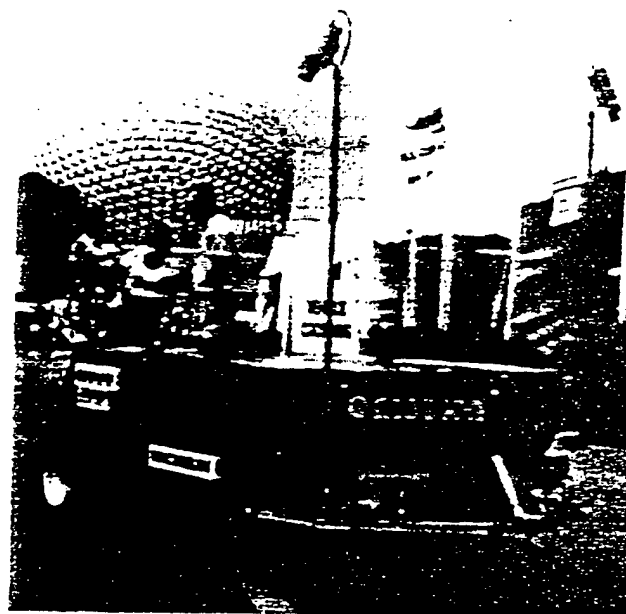


Figure 1 : Overview photo of CALVIN

This is the first year that Virginia Tech has participated in the competition. As part of our preparation, team members reviewed technical papers written by recent competitors and video tapes from the past two competitions. It became clear that the contest presents a formidable task and that a robust, well-developed and carefully tested design would be a basic requirement for success. With this in mind, the team made a concerted effort to pursue simple, reliable, cost-effective designs for the base components and subsystems. This is also the approach recommended by Gifford, et al., [1995] as part of their winning philosophy in the 1995 competition. Keeping this approach in mind, CALVIN's design incorporates many of the features that have been used successfully in recent competitions. For example, like several past entrants, CALVIN is based on the chassis of a three-wheel golf cart.

Although this platform is larger than the ideal vehicle, it does provide a rugged, stable base with ample space for equipment and payload. Also following past successes, CALVTN uses a computer vision system for line following and ultrasonic sensors for obstacle avoidance. The innovative elements of the design can be found in those areas that caused failure or were recognized as problems in otherwise successful vehicles from past competitions. These innovative features are discussed in detail later in this paper.

PROJECT ORGANIZATION

PROJECT TEAMS - As one of the largest technical universities in the country, Virginia Tech has a talented and diverse student population from which to draw. Although many large projects (such as the SAE Mini-Baja and Solar Car) have been run in the past, this autonomous robotic project generated unparalleled excitement among students and faculty. A small team of volunteers began working to organize the project in the summer of 1995. By the fall semester, the combined team working together on the two vehicles had grown to almost 50 students from across the Colleges of Engineering and Arts and Science. To direct the efforts to build CALVTN, four separate groups were formed to cover Mechanical Design, Sensing & Control, Computer Vision, and Project Organization with each group subdivided to distribute the work. To facilitate group communication, meetings were held twice a week with monthly presentations given by each group.

PROJECT TIMELINE - Organization and scheduling is a key ingredient in the success of a project of this scope and duration. Although many intermediate objectives have been achieved along the way, major project milestones are shown on the general project timeline at the end of this paper.

DESIGN PROCESS OVERVIEW

GENERAL DESIGN DESCRIPTION - The general overall design of CALVTN will be described with reference to the photographs shown as Figures 2 and 3. Figure 2 shows a top view of CALVTN with the weather-protective top cover removed. The general layout of the base vehicle, along with a number of the major components, are visible in this picture. These components include the 24 volt drive motor, timing-belt drive and differential; the 12 volt linear steering actuator with integral feedback potentiometer; the microcontroller; the on-board emergency stop (e-stops); the three 12-volt batteries and the antenna for emergency stop and radio control when not in competition. The microcontroller and electronics mount to a hinged shelf that opens to give access to the pentium-based PC and power supplies. Figure 3 is a front view of CALVTN that shows the five bumper-mounted ultrasonic sensors and the two side-viewing cameras, along with some of the weatherproof Lexan used to encase the vehicle.

DESIGN TOOLS - As with all aspects of our design, the tools used in the creative process were dictated by the overall objective of producing a safe, competitive vehicle. All Virginia Tech engineering students are trained in the use of either AutoCad or Cadkey, and both were used extensively in the design process. The uses of computer-aided design ranged from preparing a test course layout with a shape similar to the sample course shown in the contest rules, to detailed layout of the vehicle systems and subsystems. As an example, AutoCad was used to produce a layout (Figure 4) of the existing base frame of the golf cart. Attachments and modifications were then based on these drawings. Other software used during the course of the project included

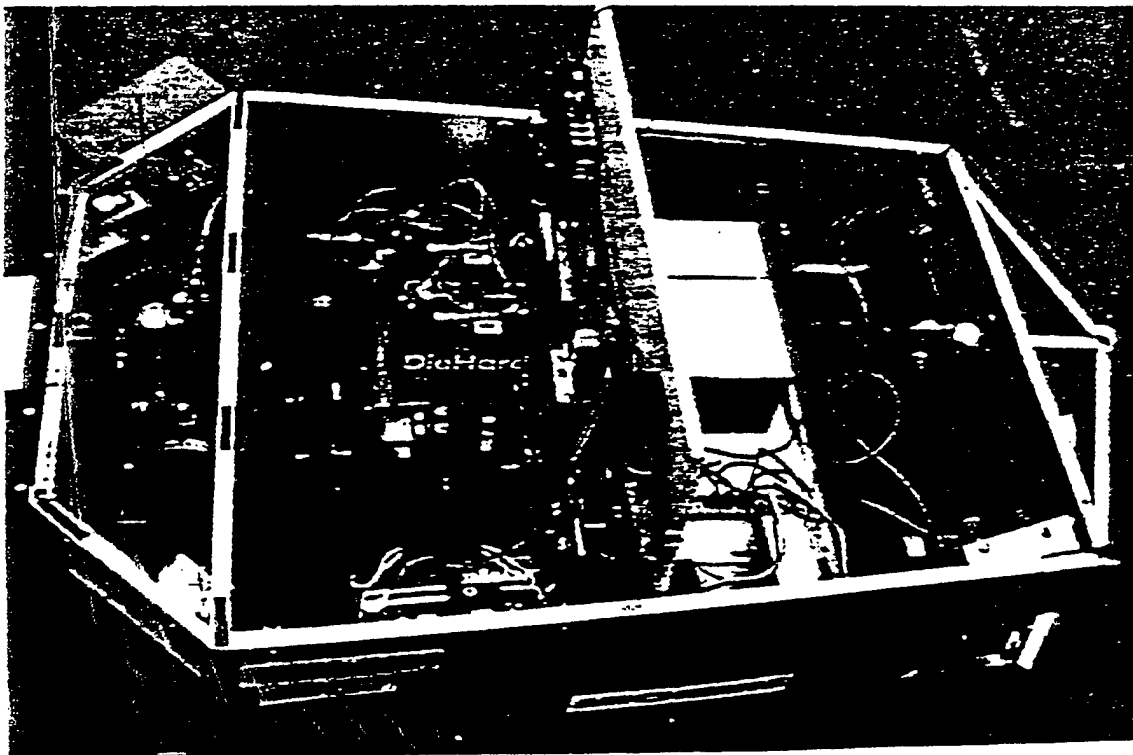


Figure 2: Top view photograph

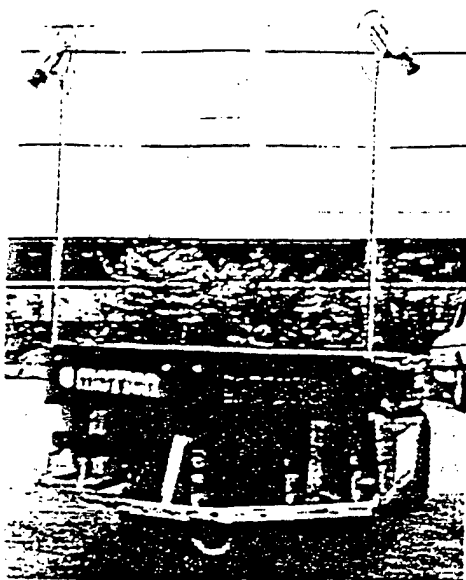


Figure 3: Front view photograph

spreadsheets, TKSolver!, MathCad and Mathematica for general mathematical modeling and MatLab for control and vibration analysis. Specialized software was also written to aid with steering kinematics and vision system modeling.

DESIGN CRITERIA (SAFETY, RELIABILITY AND DURABILITY) - The most critical design criteria were those that involved safety and those necessary to meet contest rules. Beyond these fundamental requirements, a continuing effort has been made to incorporate reliability and durability into the final design. This goal has been pursued both through the careful specification of proven, reliable components and through extensive field testing during every phase of the design. In subsequent sections, brief discussions will help to highlight the effort to produce an inherently safe, reliable vehicle. Safety was also a top priority in all shop work and during field testing.

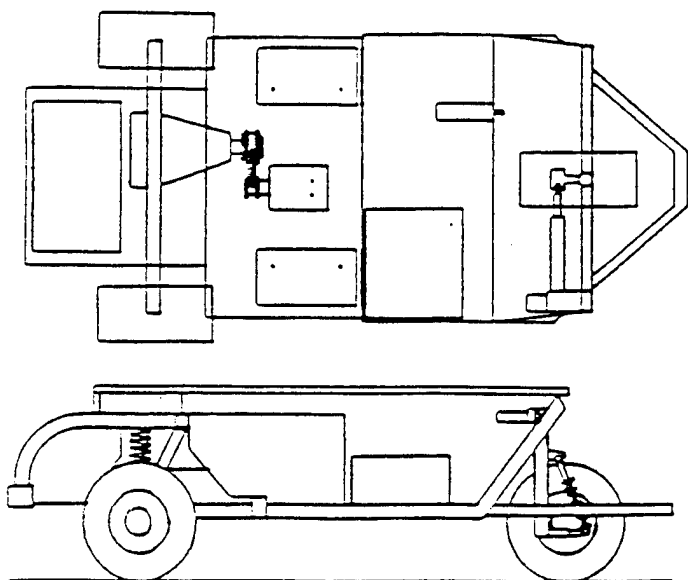


Figure 4: AutoCad Layout

FAIL-SAFE BRAKES - Acceptable braking performance was assured by using the primary components from the original golf-cart braking system. The only difference is that the actuating force normally applied by the user through the foot pedal has been replaced by a fail-safe input force from a pneumatic actuator. An electro-mechanical valve and relay system control the operation of the pneumatic actuator. While the vehicle is in operation, the main power bus closes the relay, powering the valve, thus keeping the brakes disengaged. When the circuit loses power, valve power is lost and the brakes are engaged. This condition occurs when either the onboard E-stop or the remote E-stop is activated.

STEERING ACTUATOR AND CONTROLLER - The rack-and-pinion steering system on the original golf cart was replaced with a 12-volt DC linear actuator with integrated position feedback. The motor in this system is controlled by a Pulse Width Modulation (PWM) controller. The input signal to this controller can be generated by either the microcontroller, in autonomous mode, or a radio control receiver in manual override mode. Manual override mode is used for transportation of the vehicle.

DRIVE MOTOR AND CONTROLLER - CALVIN's primary drive system uses a 0.746 kW, 24V DC motor with tachometer feedback for precise velocity control. Power is delivered to the rear wheels through a 1.8 to 1 timing belt reducer and the existing 6 to 1 golf cart differential. The speed of this motor is controlled using a PWM H-Bridge controller. The controller is capable of adjusting its PWM duty cycle so as to produce an effective continuous output voltage from -24 to +24 volts and a current of 160 amps continuous and 320 amps peak. Like the steering system, the input signal to the controller can be generated by either the microcontroller with active velocity control, in autonomous mode, or a radio control receiver in manual override mode.

ULTRASONIC AND TACTILE OBSTACLE SENSORS - CALVIN is equipped with two types of sensors in order to facilitate obstacle avoidance. First, a fan-shaped array of five ultrasonic sensors is used to locate obstacles potentially in the vehicle's path. In the event that an ultrasonic sensor detects an obstacle, an algorithm is run in order to determine how the vehicle's path should be changed. This routine considers the vehicle's distance to the obstacle, and which ultrasonic sensors in the array detected the obstacle.

The tactile sensors consist of three push-bars on the front of the vehicle. These sensors are used to indicate that the vehicle has collided with an obstacle. In the event that a tactile sensor is actuated, an interrupt routine immediately stops the vehicle and performs a reverse maneuver. The reverse maneuver steers the vehicle based upon which tactile sensor was actuated, in order to relocate the vehicle further from the obstacle. The vehicle then returns to normal operation and continues on its path around the track. The ultrasonic and tactile sensor arrays can be seen in Figure 5.

COMPUTER AND SOFTWARE - The vision system is implemented using a pentium 100MHz PC and an FF1 DSP Frame Grabber from Current Technology, Inc. All vision algorithms were developed using standard C and the frame grabber's C libraries. Standard C is also used to code and compile the software necessary to communicate between the computer and the microcontroller.

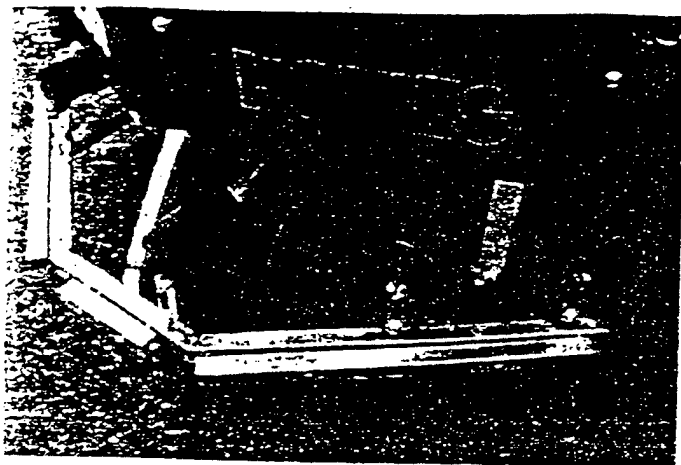


Figure 5: Ultrasonic and Tactile Sensor Arrays

The vision system is based on two cameras looking to the left and right of the vehicle, but only one camera is active at a time. When the active camera obtains a series of three consecutive poor images, the system switches to the view on the other side. Image processing is performed on a view of the course directly adjacent to the vehicle. Of this whole image, only two smaller regions of interest (ROI) are considered to be significant. All image processing is then performed on these ROIs only, in order to save computation time. The frame grabber first thresholds the gray-scale camera image to convert the image to binary black and white. The thresholding routine is dynamically adjusted for each ROI to recognize the white paint on grass as the brightest portion of the image. This means that the ROI images used to navigate consist of white line segments on an all-black background. The vision code then uses these two ROI to detect a forward and a rear segment of the course line. In order to discern the line, the routine finds the grouped blobs which it extracts as a segment of the line. The location of the centroids of these blobs are then compared to those found when initializing the vehicle in the center of the course. This information is used to determine the vehicle's distance and angle relative to the line defining the course. The superposition of this zeroth order and first order path information results in a desired steering angle. This angle is then processed to be transmitted to the microcontroller.

The microcontroller integrates the desired steering angle (based solely on vision) with information from the ultrasonic and tactile sensors. Priority must be assigned to obstacles potentially in the vehicle's path; therefore tactile and ultrasonic sensor information is, in some cases, weighted more heavily than the vision in determining the actual steering angle. Consequently, the microcontroller combines the three inputs to determine the path of the vehicle. In effect, the microcontroller performs all onboard navigation.

SENSORS, ELECTRONIC LAYOUT AND SYSTEM INTEGRATION - Figure 6 is a schematic diagram showing the general architecture and integration of the sensors, electronics and computers used on-board CALVIN. Once an image has been processed, the PC sends a steering angle to the microcontroller via a serial connection. If the course boundary line is not clearly detected by the vision system in several successive frames, the PC

also sends a command to the microcontroller to switch to the second camera, and the vision system begins tracking the opposite line. Ultrasonic signal processing as well as integration and direct control functions are handled by the Motorola 68HC11-based microcontroller. This microcontroller also executes closed-loop control of the steering and drive motors using pulse-width-modulated signals. A schematic layout of the battery connections and power control elements is shown in Figure 7.

DESIGN PROBLEMS AND SOLUTIONS - Throughout the design process for CALVIN, precise design tools and techniques were used whenever possible. Occasionally, however, an event would arise where trial-and-error methods were necessary. The first arose from the imprecise geometry of the base frame. Calvin began as a golf cart, but came with no factory dimensions. Measurements were made and CAD drawings constructed, but the angles of the frame were difficult to accurately measure, and small differences had a great effect on the positioning of integral parts. This forced a trial-and-error technique on mounting the drive motor, steering actuator and tachometer.

Electrical problems are also unavoidable on a vehicle with so many components. Floating grounds and ground loops are primary contributors to these difficulties. During the layout and wiring of the vehicle, an effort was made to keep the wires as short and organized as possible to facilitate quick troubleshooting. Still care had to be taken to reduce the interference in many of the electrical components, most notably the cameras and the monitors. To do this, ground wires were consolidated, and shielded coaxial wire was used when possible.

PERFORMANCE PREDICTIONS

BRAKING AND SPEED CONTROL - The most critical performance issues were those that involved safety requirements and meeting contest guidelines. Braking and speed control, especially on an inclined surface, are obviously of great importance. An earlier section of this paper describes the design of the braking system and how the foot pedal input was replaced by a pneumatic actuator. Acceptable performance was assured by selecting an actuator that supplied the same brake cable tension as a typical user would supply through the brake pedal in an emergency condition (roughly 500N).

Accurate speed control is important for safe vehicle operation and for assuring constant performance of the vehicle on sloped surfaces. Several past contestants reported control problems that were directly attributed to speed variations. Velocity is controlled on CALVIN using a tachometer attached directly to the output shaft of the drive motor to provide a voltage signal proportional to vehicle speed. This signal is fed back to the microcontroller, which issues a pulse-width-modulated control signal to the motor driver. By modeling the vehicle as a mass moving on a frictionless surface, control system gains can be selected to give stable velocity control with an adequate response time.

RAMP-CLIMBING CAPABILITIES AND SAND PIT PERFORMANCE - Competition rules permit ramps or inclined terrain to have a grade of up to fifteen percent. This information, coupled with the 8.04 km/hr maximum allowed speed and the vehicle's estimated weight, can be used to directly compute the minimum required power of the drive motor (since power equals force multiplied by velocity in the direction of the force). A twenty

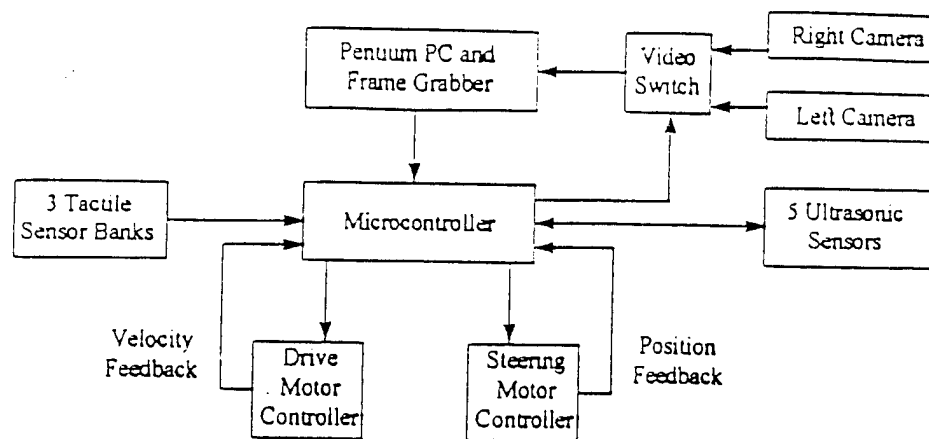


Figure 6: Sensor and control integration

percent grade and a 8.04 km/hr vehicle speed produce a vertical velocity component of 1.61 km/hr. Assuming a 1780 N vehicle weight and no friction results in a required drive motor of about 0.746 kW. Recognizing that steep inclines would be traversed at 1.61 km/hr or less, this was taken to be a conservative estimate. A 0.746 kW continuous, 1.49 kW peak, 24 volt DC drive motor was selected and purchased from a surplus catalog. Using a speed reduction of about 10:1 through a timing belt drive and the golf cart differential, the desired 8.04 km/hr peak speed and ramp-climbing ability were achieved as predicted. Designing a vehicle to traverse a sand pit was a less precise prediction. Goodyear, who donated the vehicle tires, provided assurances that these wide, soft tires would perform well in sand and would not damage the grass portion of the course.

REACTION TIMES AND LINE-TRACKING PERFORMANCE - With any digital control system, the faster the update speed of the controller, the more stable the performance of the controlled device under rapidly changing conditions. The

computer vision and ultrasonic acquisition and computation requirements obviously limit the overall update rate of the controller. The team attempted to partially address this issue during conceptual design by separate, parallel processing of the ultrasonic and vision feedback. The vision feedback processing is accomplished directly on the frame grabber card or in the host Pentium 100 MHz personal computer. Operation of the ultrasonic sensors and the associated processing are accomplished in the Motorola 68HC11-based microcontroller. This allows a maximum update rate to the steering and drive motors of about 6 Hz. At peak vehicle speed, this amounts to an update about once every 0.3 m of travel. Through testing, this has been determined to be marginal for robust line tracking; one or two bad images can result in the vehicle traveling out of bounds. Nevertheless, within the constants of the existing equipment, this is a reasonable update rate. At competition, we intended to run CALVIN at peak speeds of about 3.22 km/hr to maintain stable operation.

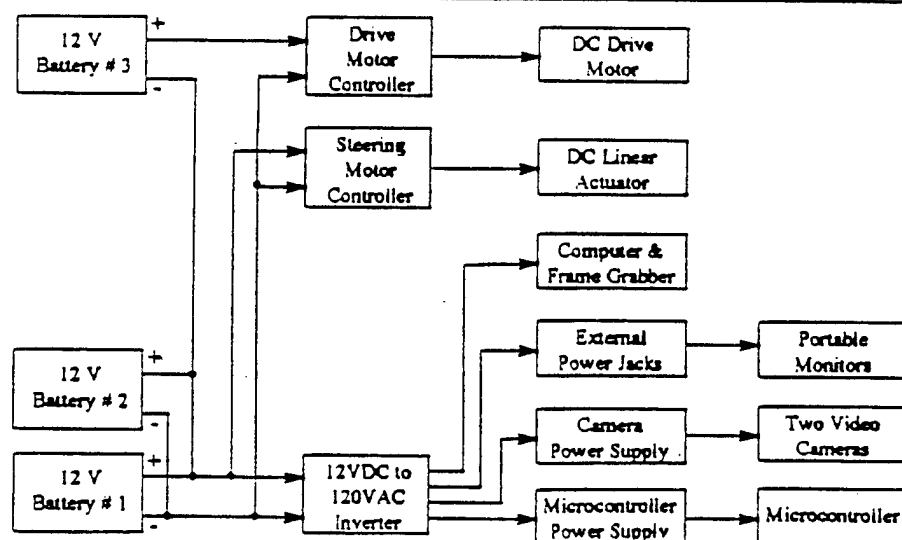


Figure 7: Power Supply diagram

INNOVATIVE ASPECTS OF THE DESIGN

The design philosophy used in developing CALVIN did not specifically include innovation as a design objective. Our primary objective was to design the most competitive possible vehicle under the constraints of the governing contest rules and limited financial resources. Nevertheless, the resulting design is believed to be innovative in several significant respects. In all cases, however, the innovative features are intended to address problems or shortcomings reported in similar vehicles from previous competitions.

A simple example of this was the design of a system that uses only three 12-volt deep-cycle batteries as the source for all on-board power. Power for the PC and peripheral equipment was supplied using an 400 Watt inverter to generate 120 volt AC power. Other DC power requirements were met through small on-board power supplies or through series connection of the batteries. This power supply arrangement was previously shown as Figure 7. While this may seem to be a minor issue, a number of previous teams, including the first-place 1995 Colorado team [Gifford, 1995], cited the maintenance of several dissimilar batteries as a significant disadvantage. The University of West Virginia team also cited the weight penalty and potential danger associated with their eight-battery, 96-volt DC bus system [Banta, 1995].

A second, and more significant, innovative feature of CALVIN is the use of two independent cameras for line tracking. This innovation is an attempt to address, in a cost-effective way, the problems noted by 1994 and 1995 competitors with single-camera systems [Murphy, 1995, Gifford, et. al., 1995]. Originally, the team hoped to use two cameras and two independent frame grabbers. With such a system, each camera could continuously track one of the two course boundary lines. Unfortunately, two frame grabber cards or a single multi-input frame grabber proved to be too expensive, and an alternative innovative solution was pursued. The end result was to use a simple relay-actuated video switch to change camera views when the course boundary lines disappeared or became difficult to detect. Using a line-integrity check based on the size and grouping of white image areas, the computer vision system makes three attempts to find a line with the current camera. If two successive attempts fail, the vision software sends a command to the microcontroller to switch to the camera on the other side of the vehicle and begin tracking the opposite line. Based on a video update rate of at least six frames per second (bad images can often be rejected quickly, resulting in a higher rate), and a maximum speed of 8.04km/hr, the vehicle will travel no further than 0.244 m in this time.

A third innovative feature of CALVIN's design is the use of a radio controller to generate the same PWM signals as the microcontroller for the steering and drive motor controllers. This allows simple switching between autonomous and manual modes, which greatly facilitates vehicle setup and testing. During normal operation, all user interaction with CALVIN is through the radio controller or through a single weather-resistant control panel mounted on the left rear quarterpanel of the vehicle. This control panel is shown in Figure 8 below. While this is again a simple feature, it is a clear improvement over the interfaces used by vehicles in previous competitions, many of which required awkward interactions including manual pushing for placement and setup.

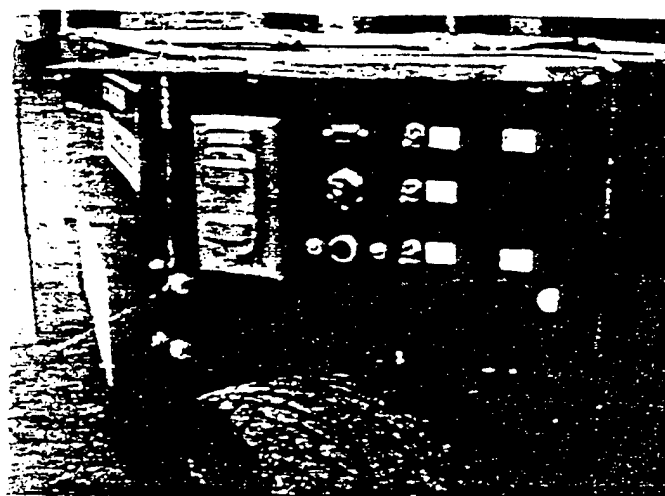


Figure 8: The control panel

A fourth innovative feature of CALVIN is the separation of data processing between the PC and the microcontroller. Since data processing speed has historically limited other vehicles maximum speeds [Cheok, 1994, Nagy et. al., 1995], a system was designed to process data in parallel. By processing closed-loop velocity and steering control, as well as ultrasonic and tactile sensor data on CALVIN's microcontroller, the PC is used solely for the computationally intensive vision algorithms. This separation of data processing greatly reduces the total processing time, therefore increasing the maximum sustainable vehicle speed.

VEHICLE COST AND FUNDING SOURCES - The table at the end of the paper lists the ready-made components that were purchased or donated. Separate columns list both the retail cost and the purchase price of the component. Note that many items were either donated or sold to the project team at a substantially reduced cost. Also note that all fabricated components, such as mounting brackets, sensor housings, framework, bumper and the Lexan outer shell, were built by members of the design team in the student shop of the Mechanical Engineering Department. In addition, team members did a great deal of electronic design and circuit fabrication. It is estimated that approximately 5000 person hours have been invested in this project. This includes time spent on activities such as project organization, fundraising, publicity, and research.

COMPETITION RESULTS

The 1996 Unmanned Ground Vehicle competition held at Walt Disney World, Epcot Center July 13-15, was composed of two sub-competitions. The first was the SAE Design Competition, which evaluated the written report, an oral presentation and a static viewing for each vehicle.

The second competition was the obstacle course race. All vehicles were given three attempts at completing the course, and the vehicle to complete the course in the shortest amount of time, or to proceed the furthest in any one run would be declared the winner of this competition.

Virginia Tech's CALVIN vehicle was declared the winner of the SAE Design Competition. Virginia Tech's other vehicle, BOB, scored third place in the SAE Design Competition. The

winner of the obstacle course race was Ohio State University's vehicle. Virginia Tech's obstacle course race results were 13th and 6th for CALVIN and BOB respectively.

ACKNOWLEDGMENTS

Thanks to the many corporate sponsors and to the Departments of Electrical and Mechanical Engineering for making this project possible. Special thanks to Mr. William Agnew of SAE for carefully reviewing this document before final submission.

Part Name and Description	Project Cost (\$)	Retail Cost (\$)
Base Vehicle - Gas Powered E-Z Go golf cart (used)	500	500
Drive Train - 12 VDC 0.764 kw Thermo King Motor (surplus)	100	100
Drive Train Components (belts, pulleys, etc.)	75	75
Computer - Pentium 100 Mhz PC	1125	1125
DSP Frame Grabber (Current Technologies)	donated	900
Motorola 68HC11-based Microcontroller (Coactive Aesthetics)	125	250
Assorted Electronic Components and Cables	150	150
Polaroid Ultrasonic Sensors (six)	350	350
Linear Actuator - 12 VDC 6" stroke (Motion Systems)	donated	300
2 Cameras & Lenses (DEI / Professional Security Alliance)	400	1200
PWM Motor Controllers (VANTEC)	450	950
Aluminum for Frame and other structures (R.J. Reynolds)	donated	200
Batteries (Sears)	150	300
Air Tank (Steel-Fab)	donated	100
400 W Inverter (Tripp-Lite)	donated	100
Lexan for Outer Shell (Piedmont Plastics)	donated	350
TOTAL	3,425	6,950

Vehicle Cost Breakdown

May 1995	Aug. 1995	Oct. 1995	Dec. 1995	Jan. 1996	Mar. 1996	May 1996	July 1996
Autonomous Robotic Vehicle Project Introduction at Virginia Tech Preliminary design team assembled	Golf cart acquired for base vehicle Academic year begins, large project team organized	Base vehicle operational and prepared for modification	Conceptual design/frame modifications completed Fail-safe brake system implemented	Spring semester teams organized System components ordered Drive and steering systems tested	Computer vision and ultrasonic sensors tested Automatic drive control specified	Formation of summer teams Systems integration completed Testing and subsequent modification begins	Inter-team competition between CALVIN and BOB Design paper submitted Departure for Epcot 7/12 4th Annual UGR competition in Epcot

Project Timeline

REFERENCES

Murphy, Robin R., "An Artificial Intelligence Approach to the 1994 AUVS Unmanned Ground Robotics Competition," IEEE International Conference on Systems, Man, and Cybernetics, Oct., 1995, Vancouver, B.C., Paper 0-7803-2559-1/95.

Gifford, K. K., Deeds, M., Van der Hoek, A., Henning, F., Freeman, J., Haussman, G., Kuzminsky, S., Stoller, S., "RoboCar: Software, Hardware, and Mechanical Design Issues for the University of Colorado Autonomous Rover Vehicle," SPIE Vol. 2591, Sept., 1995.

Banta, L. E., "Undergraduate Robot Design at West Virginia University," Mobile Robots X, SPIE, Vol. 2591, 1995, pp. 202-208.

Cheek, K. C., "Autonomous Unmanned Ground Robotic Vehicle Competition: An Intelligent Control Challenge," Proceedings of the American Control Conference, Vol. 1, 1994, pp. 383-387.

Gifford, K. K., Deeds, M., van der Hoek, A., Henning, F., Freeman, J., Haussman, G., Kruzminsky, S. and Stoller, S., "RoboCar: Software, Hardware, and Mechanical Design Issues for the University of Colorado Autonomous Rover Vehicle," Mobile Robots X, SPIE, Vol. 2591, 1995, pp. 228-238.

Matthews, B., Ruthemeyer, M., Perdue, D. and Hall, E. L., "Development of a Mobile Robot for the 1995 AUVS Competition," Mobile Robots X, SPIE, Vol. 2591, 1995, pp. 194-201.

Murphy, R. R., Hoff, W., Blitch, J., Gough, V., Hawkins, D., Hoffman, J. C., Krosley, R., Lyons, T., Mali, A., MacMillan, J. and Warshawsky, S., "Colorado School of Mines Behavioral Approach to the 1995 UGR Competition," Mobile Robots X, SPIE, 1383, November 1990, pp. 436-447.

Nagy, P. V. and Bock, T., "The Northern Illinois University Autonomous Mobile Robots," Mobile Robots X, SPIE, Vol. 2591, 1995, pp. 209-219.

Navigation of an autonomous ground vehicle using the subsumption architecture

Paul J. Johnson, Kevin L. Chapman, John S. Bay

Machine Intelligence Laboratory, Bradley Department of Electrical Engineering,
Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0111

ABSTRACT

The subsumption architecture is used to provide an autonomous vehicle with the means to stay within the boundaries of a course while avoiding obstacles. A three-layered network has been devised incorporating computer vision, ultrasonic ranging, and tactile sensing. The computer vision system operates at the lowest level of the network to generate a preliminary vehicle heading based upon detected course boundaries. The network's next level performs long-range obstacle detection using an array of ultrasonic sensors. The range map created by these sensors is used to augment the preliminary heading. At the highest level, tactile sensors are used for short-range obstacle detection and serve as an emergency response to obstacle collisions. The computer vision subsystem is implemented on a personal computer, while both ranging systems reside on a microcontroller. Sensor fusion within a subsumption framework is also executed on the microcontroller. The resulting outputs of the subsumption network are actuator commands to control steering and propulsion motors. The major contribution of this paper is as a case study of the application of the subsumption architecture to the design of an autonomous ground vehicle.

Keywords: control systems, subsumption architecture, sensor fusion, autonomous vehicles, obstacle course navigation, computer vision, ultrasonic ranging

1. INTRODUCTION

This paper describes the control system design issues encountered by the Virginia Tech Autonomous Vehicle Team in its development of two autonomous ground vehicles. These vehicles were built for entry in the Fourth Annual International Autonomous Ground Robotics Vehicle Competition, which was sponsored by the Association of Unmanned Vehicle Systems International (AUVSI) and was held at the Epcot Center in Orlando, Florida on July 13-15, 1996. The goal of the competition was to have an unmanned ground vehicle navigate an outdoor obstacle course roughly 500 feet long in no more than ten minutes, while obeying a speed limit of five miles per hour. White and yellow lane markers were painted on the ground to define the course's boundaries, while hay bales served as obstacles^{9, 10, 11}.

Successful obstacle course navigation requires the ability to detect course boundaries and obstacles, and then make intelligent control decisions. While it would be possible to gather all needed boundary and obstacle information using a single sensory device, such as a camera attached to a computer vision system, it is more likely that multiple types of sensors would be used. This would allow the vehicle to utilize the abilities of very effective, but highly specific, sensory devices. The problem that then arises is that of combining these different sensing modalities into a limited number of control signals. This is known as the *sensor fusion problem*^{5, 6, 13}.

One solution to the sensor fusion problem can be found in the *subsumption architecture*^{1, 2}. This architecture decomposes a control problem into a collection of behaviors. Individual behavior modules are developed to process sensory data so as to create a simple behavior, such as line following. Multiple modules, each with potentially different sensory inputs, are then developed to produce additional behaviors. Through the use of a hierarchy, outputs of some modules are allowed to subsume the outputs of other modules, thus creating a single actuator command from multiple sensory inputs.

The focus of this paper is the use of the subsumption architecture to perform sensor fusion for the control of autonomous ground vehicles. General design issues for autonomous navigation of an obstacle course are first discussed, with an emphasis on sensor fusion. A brief summary of the subsumption architecture is then given, with

*SPIE Int'l. Symp. on Intel. Systems and Manufacturing,
Nov. 1996. pp 54-62.*

a description of the implementation of the subsumption architecture on two different vehicles. A discussion of future areas of work is also included.

2. AUTONOMOUS NAVIGATION ISSUES

2.1 Information needed for obstacle course navigation

Many issues are involved in designing an autonomous vehicle to navigate an obstacle course of the type established by the AUVSI for the vehicle competition. These issues can be divided into various *functions*, such as gathering all necessary sensor data, processing sensor data, and applying control signals to the vehicle's actuators, or they can be divided into various *behaviors*, such as path following and obstacle avoidance^{9, 10, 11}.

No matter how the issues are divided, the vehicle must be able to detect the course's boundaries and the multiple obstacles it may encounter. How the boundaries are defined and what one wishes the vehicle to do with boundary information dictate which technologies are candidates for boundary detection. Computer vision can detect boundary lines both near and far, providing the vehicle with information about its present location and its possible future locations. An infrared emitter/detector pair can also be used for boundary detection. This method relies on the reflection of an emitted signal and thus the emitter/detector must be in close proximity to the line. Using this method, it would be very difficult to get any knowledge about where the vehicle should go in the distant future.

The vehicle must also be able to detect obstacles. Ideally, proper detection should allow obstacles to be avoided altogether. Detecting obstacles from a longer distance is beneficial, as more time is available to take evasive action. Computer vision systems can be used to detect the location of obstacles at significant distances, but extensive image processing may be required for both obstacle recognition and distance calculations to the identified obstacle. A simpler approach is to use soundwaves from ultrasonic transducers to locate obstacles. These sensors have nice distance detection capabilities, but because they tend to have limited fields of view, an array of ultrasonic sensors is usually needed to provide sufficient coverage^{4, 5, 9, 11}.

In the 1996 competition, collisions with an obstacle were acceptable, but moving an obstacle was not. Tactile sensors can be used to provide information about collisions. As with the ultrasonic sensors, an array of tactile sensors should be used to determine where on the vehicle the collision occurred. This allows corrective measures to be taken to avoid hitting the obstacle a second time.

2.2 Sensor fusion

Autonomous navigation most likely will involve gathering and processing data from a variety of sensor types. In the previous section, computer vision, infrared emitter/detector pairs, and ultrasonic and tactile sensors were discussed in relation to detecting boundary lines and obstacles in the context of obstacle course navigation. In general, laser rangefinders, inclinometers, photosensitive resistors, Global Positioning Systems (GPSs), and many more devices may be used to provide information to a vehicle. Combining all of this information in an intelligent and organized fashion is called sensor fusion.

It is conceivable that sensor fusion could be avoided altogether in obstacle course navigation by using only computer vision. However, there are numerous disadvantages to this approach:

- Computer vision is not trivial, as complex image processing can be computationally intensive
- Other sensing technologies may be better at performing a few specific tasks, such as tactile sensors for obstacle collision detection
- Diverse and unpredictable lighting conditions diminish the reliability of a computer vision system
- In general, using a single sensing modality provides a single point of failure

A system capable of performing sensor fusion addresses all of the points listed above. Perhaps the biggest advantage of such a system is that sensor multiplicity can provide redundancy. Sensor uncertainty and error will

exist in autonomous vehicle navigation^{4, 5, 8, 11, 12}. Redundancy helps to lessen the harmful effects of sensor uncertainty and error.

Another advantage of sensor fusion is its inherent degree of modularity. Creating a system that is modular has many benefits in both the initial design stage and later refinements. Ideally, as more sensor information becomes available, this information should be able to fuse with existing data easily, without requiring large changes to current software and hardware. Sensor fusion techniques are based on this ability to combine multiple modules into a single system.

2.3 Reactive systems versus planning systems

The actions of an autonomous vehicle can be the result of two general types of control structures: *reactive* or *planning*. Both of these control structures use sensory information to determine actions. A planning control strategy assumes that the vehicle has a world model on which to base all of its actions. Information is sensed, processed based on the world model, and then an action is generated. The world model may be known *a priori*, or alternatively, may be developed from information gathered by the vehicle's sensory inputs. A reactive system differs in that it uses no world models and its sensory inputs are subjected to minimal processing before generating outputs, leading to a quick response from a given stimulus.

In most cases of autonomous navigation, a detailed world model will not be known *a priori*. If a planning system is to be used, a model of the world must be built from sensory inputs. Unfortunately, sensory systems can typically only provide a partial picture of the environment^{4, 6}. Sensory information gathered is also inherently noisy, as sensors may not be operating at all times under ideal conditions. This can lead to inaccurate world models and poor performance in a planning system.

Reactive, or behavior-based, systems do not require the development of a world model and are relatively simple compared to planning systems. Participants in previous competitions have proposed using reactive systems due in part to the relative simplicity of such designs^{10, 11}. The behavior-based approach is also ideally suited to obstacle course navigation, as the task can be decomposed into distinct behaviors such as path following, line following, obstacle avoidance, etc.

3. SUBSUMPTION ARCHITECTURE OVERVIEW

The subsumption architecture is a behavior-based control scheme developed for controlling autonomous mobile robots^{1, 2}. Control of an autonomous mobile robot will inevitably involve multiple goals, some of which may require conflicting actuator demands. Any successful architecture must have a means of arbitrating these conflicting demands. The subsumption architecture does this by taking a behavior-based approach to decomposing the entire control problem.

A typical approach, as shown in Figure 3.1, is to break down the problem into a series of functional units, where each unit performs a specific function. For an autonomous vehicle to navigate an obstacle course, these functions may include gathering sensor data, processing sensor data, making logical steering and drive decisions, and applying control commands to steering and drive actuators. There are, however, multiple disadvantages to this type of architecture:

- The details of every module must be considered before the construction of any individual module can proceed.
- Module interfaces are extremely important because of the module-to-module flow of information.
- A problem in one module can lead to a cascade failure in other modules that depend on the output of the damaged module.

The subsumption architecture takes an alternate approach by decomposing the problem into parallel, task-achieving behaviors, as shown in Figure 3.2. Individual behaviors are generated from modules, each of which is a simple, asynchronous computational machine. An entire control system is then constructed using layers of these

behavior modules. The vehicle achieves a certain level of competence for each layer in its control system. As more modules are added, the overall level of competence of the vehicle increases. Higher layers produce more specific desired behaviors and can subsume lower layers by suppressing the lower layers' outputs. For autonomous obstacle course navigation, these behaviors may include path following, obstacle avoidance, and reaction to obstacle collisions. This style of architecture addresses all of the problems associated with the series architecture of Figure 3.1: modules are constructed individually, module interfacing is not elaborate, and failures in one module have a minimal affect on other modules.

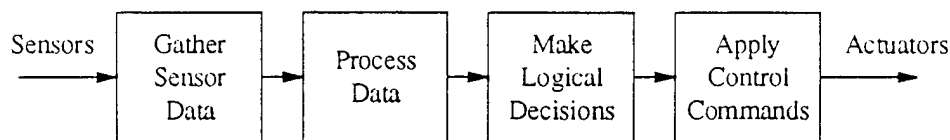


Figure 3.1 A series of functional modules for obstacle course navigation

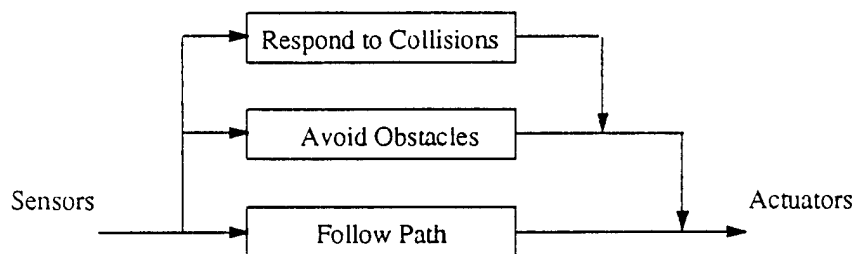


Figure 3.2 Parallel behavior modules for obstacle course navigation

The subsumption architecture essentially involves fixed priority assignments between behaviors. The commands belonging to the highest priority behavior are applied to the robot's actuators. The designer of such a system must decide on the number of behavior levels and what priority to assign to each level. The ordering of the modules in Figure 3.2 shows one possible behavior decomposition and hierarchy for the obstacle course navigation problem.

4. VEHICLE DESIGNS

Past competitors have designed vehicles using computer vision to detect the course's boundaries and ultrasonic sensors to detect the hay bale obstacles^{9, 10, 11}. The Virginia Tech Autonomous Vehicle Team used a similar approach in designing two vehicles, BOB (Beast of Burden) and CALVIN (Computerized Autonomous Land Vehicle with Intelligent Navigation).

Figure 4.1 provides an overhead view of the two vehicles, showing their relative sizes and the positions of the cameras and ultrasonic sensors. BOB used a personal computer with a frame grabber card to interface with a single camera, which was positioned near the middle of the vehicle and directed forward. CALVIN was equipped with two cameras, each looking directly out to opposite sides of the vehicle. These cameras were positioned near the front of the vehicle. An electronic video switch was used to connect either one of these two cameras to a frame grabber card in a personal computer. Both vehicles used a Motorola 68HC11-based microcontroller to interface to an array of five ultrasonic sensors and to three tactile sensor banks. The positions of the ultrasonic sensors were different for each vehicle due to the different vehicle widths and the different bumpers on each vehicle. The three tactile sensor banks were connected to the three segments of the front bumpers on each vehicle.

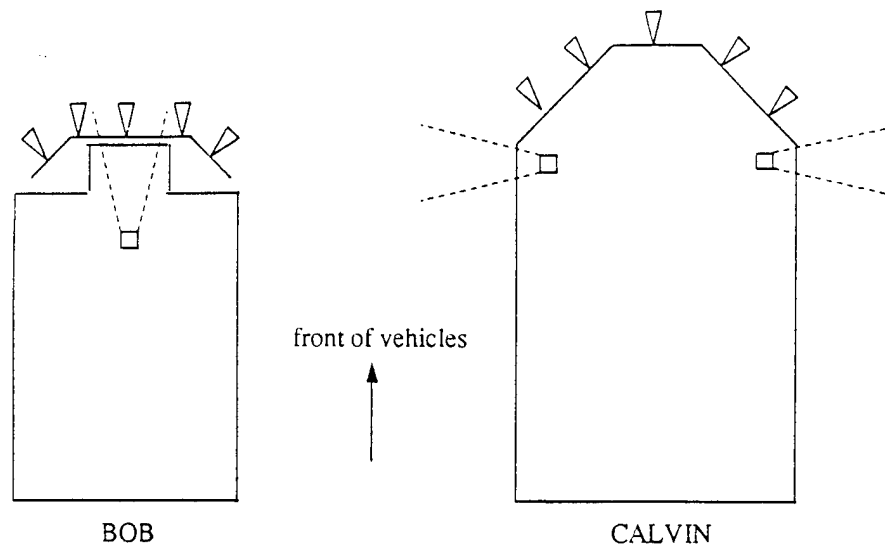


Figure 4.1 Top view showing relative sizes and configurations of the two vehicles

Figure 4.2 shows a block diagram of the control systems for the Virginia Tech autonomous vehicles. Three different sensing modalities were combined into reference signals using the subsumption architecture to control a steering motor and a drive motor. Sensor fusion is inherent in the subsumption architecture, where outputs resulting from one sensor can be subsumed by outputs resulting from another sensor. A previous competitor⁹ has stated that obstacle course navigation is ideally suited to subsumption architecture because the task can be broken down into behaviors such as stay within the lines, follow a specific line, and avoid obstacles.

Figure 4.3 shows the sensor fusion block of Figure 4.2. The bottom layer was the Path Following module, which used computer vision to create a path following behavior. The middle layer was the Obstacle Avoidance module, which used ultrasonic sensors to avoid obstacles. At the highest level was the Emergency Obstacle Avoidance module, where tactile sensors were used to trigger an emergency response if an obstacle was hit.

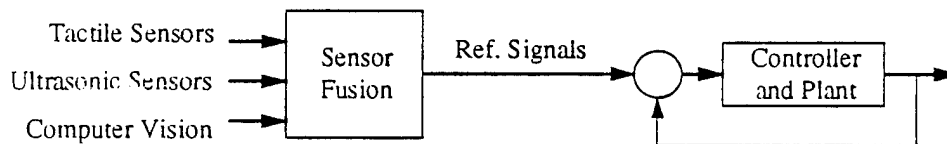


Figure 4.2 Autonomous vehicle control system

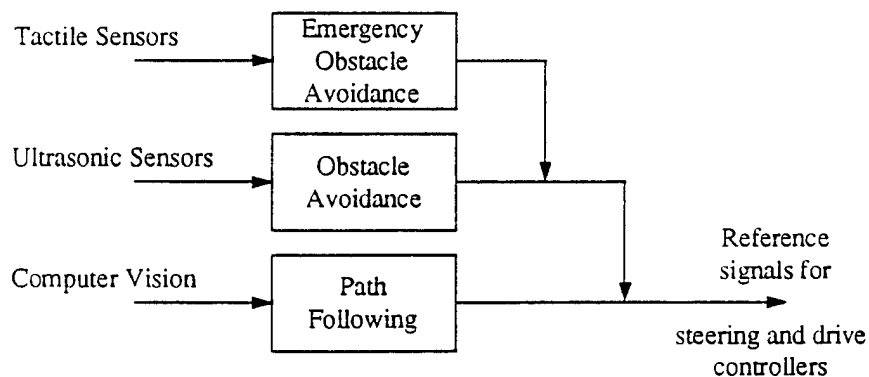


Figure 4.3 Subsumption architecture approach to sensor fusion

Sensor fusion was accomplished on the microcontroller. The Path Following module was realized on the personal computer, with its output sent to the microcontroller via a serial connection. The Obstacle Avoidance and Emergency Obstacle Avoidance modules operated solely on the microcontroller. Software on the microcontroller allowed the middle layer to subsume the lowest layer. The interrupt capabilities inherent in the hardware of the microcontroller were used to allow the highest layer to subsume the two lower layers' outputs. The next three subsections of this chapter provide additional details about each behavior module.

Figures 4.4 and 4.5 show schematic diagrams of the integration of the personal computer and microcontroller with the sensors and controllers for the two vehicles. BOB's drive motor controller performed closed-loop velocity control, eliminating the need for any velocity feedback to the microcontroller in Figure 4.5.

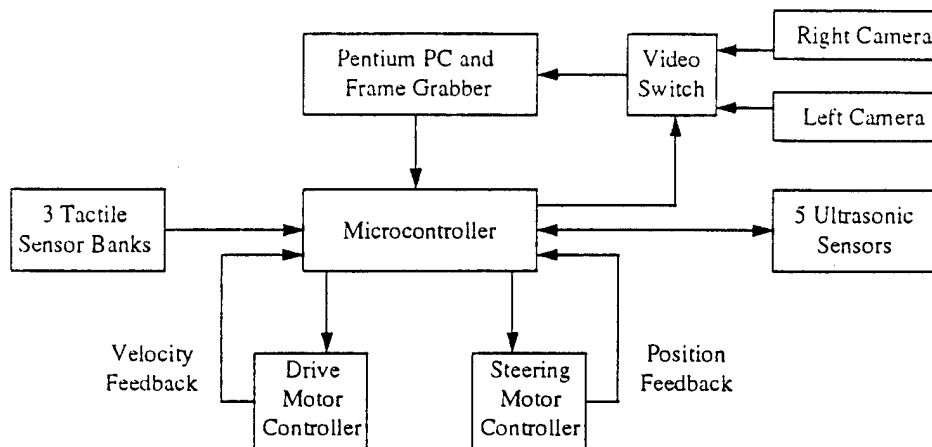


Figure 4.4 Computer integration of sensing and control for CALVIN

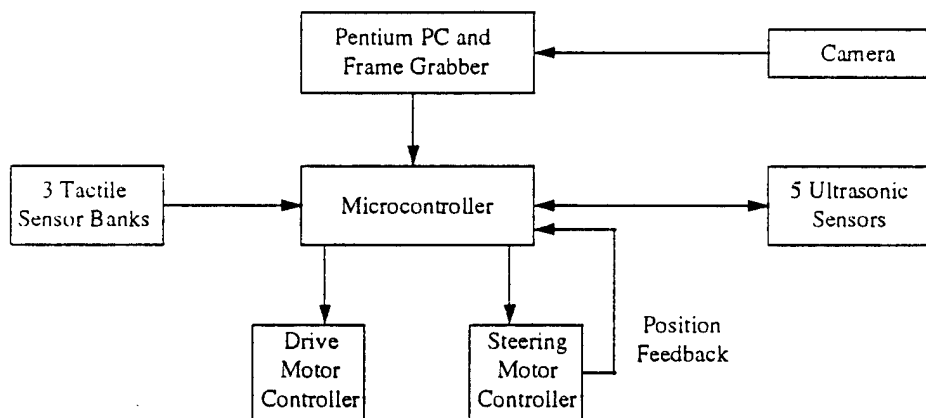


Figure 4.5 Computer integration of sensing and control for BOB

4.1 Boundary detection through computer vision: Path Following module

On both vehicles, the input to this module was a camera image and the output was a preliminary steering angle for keeping the vehicle within the boundaries of the course. The internal operations of this module, however, were significantly different between vehicles because each used a different camera configuration.

On BOB, there was just a single camera directed straight ahead. Since one would expect boundary lines to extend up the image plane, a weighted sum of pixels in each column was taken, with the goal being to find two modes in

this column data. These modes, representing an estimate of the boundary locations, were used to create a preliminary steering angle that moved the vehicle toward the horizontal average of the two modes. The horizontal average approximated the center of the course.

On CALVIN, two cameras were used, each directed outward at a right angle to the vehicle's forward direction. One would expect boundary lines to appear as horizontal lines at a certain vertical position in these images. If the detected line was not horizontal, CALVIN was not parallel to the boundary line. Similarly, if the line was horizontal but not at the correct vertical position, then CALVIN was either too close or too far away from the boundary. A steering angle was chosen to properly orient the vehicle based upon the angle and vertical location of the detected line. If consecutive images were bad from one camera, a signal was sent to the microcontroller to have it switch cameras, providing a degree of redundancy.

4.2 Obstacle detection through ultrasonic sensors: Obstacle Avoidance module

On both vehicles, five ultrasonic sensors located in the front of the vehicle were used to gather distance data to the obstacles. A collection of IF-THEN rules was used to analyze the distance values returned by the sensors. These rules determined the steering angle needed to avoid any obstacles in the vehicle's path. On BOB, the output of this module was a steering angle that absolutely replaced (subsumed) the preliminary heading from the Path Following module. On CALVIN, the output was either an absolute steering angle or a "steering delta" that was added to the preliminary heading from the Path Following module. The resulting sum would then be the final steering angle. This part of the sensor fusion was handled in software by the microcontroller.

4.3 Obstacle detection through tactile sensors: Emergency Obstacle Avoidance module

Tactile sensors placed on the front bumpers of both vehicles provided obstacle collision detection. Using input capture pins on the microcontroller, interrupt service routines (ISRs) were executed in response to a signal from any of three banks of wired-OR tactile sensors. These ISRs effectively subsumed the outputs of the Path Following and Obstacle Avoidance modules by forcing the vehicle to execute a "backing up" maneuver. This part of the sensor fusion was handled easily because of the ISR capabilities associated with the input capture pins of the microcontroller.

4.4 Summary of the vehicles' performance

The subsumption architecture proved to be a good method for achieving sensor fusion. The inherent modularity of this architecture made it an ideal choice for Virginia Tech's initial experiences in developing autonomous ground vehicles. The various behavior modules were developed, tested, and refined independently, greatly reducing the effort required to fully integrate all sensor systems of the autonomous vehicles. At the competition in July, BOB and CALVIN placed 6th and 12th, respectively, out of 17 participants.

The biggest problems encountered in performance trials were presented by shadows and glare from the sun. The computer vision system on each vehicle had difficulty extracting the boundary lines from a captured image when these artifacts were present. In both circumstances, the image preprocessing required could not sufficiently remove the artifact while tracking the boundary lines. Both vehicles performed well when the sun was not shining brightly. In this respect, the subsumption architecture was shown to work when the Path Following module could successfully preprocess the captured images. This also emphasizes the importance of the bottom layer in a subsumption architecture. This behavior should be the result of a very robust and fully operational module.

5. FUTURE WORK

The control systems on the vehicles performed well, but modifications are already being considered to improve the overall performance of the vehicles. Future work will focus on three areas: 1) refining the three individual modules of the subsumption network shown in Figure 4.3, 2) modifying the sensor inputs to these modules, and 3) adding additional modules to the network.

5.1 Modifications within current modules

Within the Path Following module, modifications have been proposed to use more sophisticated computer vision algorithms for line detection. The use of artificial neural networks (ANNs) for pattern recognition is one option being explored. ANNs have the ability to extract underlying geometric shapes from a noisy image, providing a degree of disturbance rejection that could counter the harmful effects of shadows and sun glare. Gathering training data that is truly representative of the numerous lighting conditions that may be encountered in competition could pose a problem for this approach.

The collection of IF-THEN rules used within the Obstacle Avoidance module could also be modified. Rather than using a large collection of crisp rules, a fuzzy rule-base has been proposed. The output of this system would still be a steering delta or an absolute steering angle, but only a small number of fuzzy rules would be used. From this small fuzzy system, a wide variety of responses would be generated. The possibility also exists to train a system to learn a set of fuzzy rules that can most effectively handle the situations encountered in obstacle course navigation.

5.2 Sensory device modifications

Alternate sensory devices for the behavior modules of Figure 4.3 are also being considered. The use of a color camera to provide the input to the Path Following module is being considered to address the problems associated with sun glare. Grass *without* any paint was seen to reflect sunlight as much as blades of grass that had been painted white (or similarly covered with lime powder). With a color camera, the intense green reflections could be filtered out while leaving the intense white reflections from the lines.

5.3 Additional modules

Line detection in an outdoor setting using computer vision is not a trivial task. It proved to be the system component most prone to failure in testing and performance trials. Therefore, the addition of a second module for boundary detection is planned. The detector used with this module would function at short range and would serve as an indication that the vehicle is very close to a course boundary. The position of the Emergency Path Following module in Figure 5.1 shows the relative priority of the behavior generated from this module.

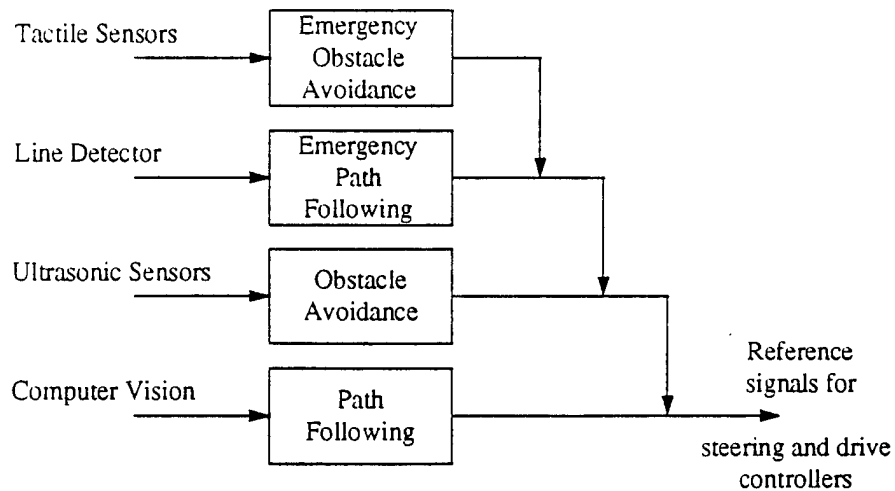


Figure 5.1 Subsumption architecture with additional module providing redundant line detection

6. ACKNOWLEDGMENTS

This paper describes the authors' efforts as members of the Virginia Tech Autonomous Vehicle Team. This team consisted of students and faculty advisors from the Mechanical and Electrical Engineering Departments, as well as

the Computer Science Department. In all, over 50 students participated in Virginia Tech's inaugural entry in the autonomous vehicle competition. The authors wish to acknowledge the efforts of everyone associated with this project. For further information, please see the Virginia Tech Autonomous Vehicle Team's home page at <http://fbox.vt.edu:10021/org/ANRobotics/robotics.html>. Information about AUVSI, the competition's sponsor, can be found at <http://avdil.gtri.gatech.edu/AUVS/index.html>.

7. REFERENCES

- [1] Brooks, R. A., "A Robot that Walks: Emergent Behaviors from a Carefully Evolved Network," *IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, May 1989, pp. 692-696.
- [2] Brooks, R. A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, March 1986, pp. 14-23.
- [3] Gasparetto, A., Rossi, A., Robb, I. A., "Control System Design and Dynamic Simulation of an Autonomous Vehicle for Factory Automation," *Proceedings of the 20th International Conference on Industrial Electronics, Control, and Instrumentation*, Bologna, Italy, September 1994, pp. 1111-1116.
- [4] Gourley, C., Trivedi, M., "Sensor Based Obstacle Avoidance and Mapping for Fast Mobile Robots," *IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 1306-1311.
- [5] Kang, D., et. al., "Position Estimation for Mobile Robot Using Sensor Fusion," *Proceedings of the 1994 International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Las Vegas, NV, October 1994, pp. 647-652.
- [6] Luo, R. C., Kay, M. G., Lee, W. G., "Future Trends in Multisensor Integration and Fusion," *1994 IEEE International Symposium on Industrial Electronics*, Santiago, Chile, May 1994, pp. 7-12.
- [7] Langer, D., Rosenblatt, J.K., Hebert, M., "An Integrated System for Autonomous Off-Road Navigation," *IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 414-419.
- [8] Lindner, J., Murphy, R. R., Nitz, E., "Learning the Expected Utility of Sensors and Algorithms," *Proceedings of the 1994 International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Las Vegas, NV, October 1994, pp. 583-590.
- [9] Matthews, B., et. al., "Development of a Mobile Robot for the 1995 AUVS Competition," *Mobile Robots X*, SPIE 2591, Philadelphia, PA, October 1995, pp. 194-201.
- [10] Murphy, R. R., "An Artificial Intelligence Approach to the 1994 AUVS Unmanned Ground Robotics Competition," *IEEE International Conference on Systems, Man, and Cybernetics*, Vancouver, B.C., October 1995, pp. 1723-1728.
- [11] Murphy, R. R., et. al., "Colorado School of Mines Behavioral Approach to the 1995 UGR Competition," *Mobile Robots X*, SPIE 2591, Philadelphia, PA, October 1995, pp. 220-227.
- [12] Niizuma, M., et. al., "Action-oriented sensor data integration and its application to control of an autonomous vehicle," *Proceedings of the 1994 International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Las Vegas, NV, October 1994, pp. 175-182.
- [13] Rich, E., Knight, K., *Artificial Intelligence*, McGraw-Hill, Inc., New York, 1991.
- [14] Yagi, Y., Okumura, H., Yachida, M., "Multiple Visual Sensing System for Mobile Robot," *IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 1679-1684.

The Distributed Learning Classifier System

Doug G. Gaff
Spatial Positioning Systems, Inc.
1700 Kraft Drive, Suite 1200
Blacksburg, Virginia 24060
dgaff@vt.edu

John S. Bay
Virginia Polytechnic Institute and State University
Department of Electrical Engineering, 340 Whittemore Hall
Blacksburg, Virginia 24061-0111
bay@vt.edu

Keywords

Distributed Systems, Network Communication, Learning
Classifier Systems, Distributed Artificial Intelligence,
Robotics

Abstract

In this paper, we present a new distributed artificial intelligence (DAI) architecture we call the Distributed Learning Classifier System (DLCS). The DLCS is an extension of the learning classifier system (LCS), with specific architecture additions for network message passing. In order to illustrate the effectiveness of the DLCS paradigm in multiple-agent scenarios, we provide a solution to the multiple-agent animat problem using the DLCS.

I. Introduction

The term "machine learning" is used to describe a vast array of algorithms, methods, and paradigms which attempt to solve tasks that run the gamut from speech recognition to financial analysis to robot control. Machine learning paradigms tend to fall into one of three categories: neural modeling, symbolic concept acquisition, or domain-specific learning [1]. The neuron-like networks present in neural modeling techniques provide fast, relatively simple mappings from input to output. Symbolic concept acquisition encompasses a broad range of artificial intelligence methods in which a machine solves a task by learning previously unconnected concepts using a predefined symbolic notation. Domain-specific learning involves a large amount of *a priori* knowledge about the task, and these algorithms focus on using an extensive knowledge-base to solve the problem [2]. In attempting to solve a particular task, one would often prefer an approach that provides a compromise between these three techniques. The *Learning Classifier System (LCS)*, proposed by John Holland [3,4], provides such a compromise.

The LCS is a rule-based, message-passing, machine learning paradigm designed to process task-environment stimuli, much like the input-to-output mapping provided by a neural network. In addition to neural-like mapping, the LCS provides learning through genetic and evolutionary adaptation to changing task environments. LCS concepts are "subsymbolic," meaning that they are encoded by the system itself and not the designer. The LCS can still be programmed with specific domain knowledge, however, and this duality provides much design flexibility [5].

The purpose of this paper is to introduce the *Distributed Learning Classifier System (DLCS)* architecture as a method for using the LCS in a multiple-agent setting. As an extension of the traditional learning classifier system, the DLCS provides a set of rules for interfacing to a standard network so that multiple LCS agents can work collectively, while still executing individually. Collective task solution is at the heart of distributed artificial intelligence (DAI) research [6]. However, the DLCS differs from other DAI architectures in that it requires no central control, and is therefore more suited for tasks involving multiple, autonomous agents. Also, since the DLCS uses a network-like message-passing scheme, standard network protocols can be used to connect agents. We begin our discussion by providing a very brief overview of the traditional learning classifier system and the current research in the area. We then present a detailed discussion of the DLCS and illustrate its application to a multiple-agent problem.

II. Background

The learning classifier system consists of a list of rules or *classifiers* that provide a set of possible actions for a given problem scenario. Each rule has one or more *condition* words and an *action* word. The system operates by reading messages from a task *environment interface*, comparing those messages to the conditions of the rules in

the *classifier list*, and posting the corresponding action messages back to the task environment. This sequence of operations is referred to as an *execution cycle* and is repeated until a particular environment state is reached. Messages from and to the environment are stored on a *message board*. Since the message board is of finite length, rules are probabilistically selected using a bidding process based on a figure of merit called *strength*. Each rule has a strength assigned to it, and rules whose conditions match the environment messages are given a *bid* value based on this strength. Rule strengths are adjusted by the task *environment payoff function* based on the appropriateness of the rule to solving the task at hand. Rule strengths are also adjusted by the *Bucket Brigade Algorithm (BBA)*, an algorithm designed to encourage rule chaining. Rule chains form when an action message from the previous iteration of the execution cycle causes an action to get selected on the current iteration. In order to provide a method for exploring new rules, the rules in the classifier list are periodically modified by a *genetic algorithm (GA)* which employs the *reproduction* and *mutation* genetic operators. The combination of the GA (*rule discovery*) and rule strength adjustment (*credit assignment*) enable the LCS to learn new concepts. Credit assignment and rule discovery are performed after actions are posted to the environment. For a more detailed discussion of the LCS, see [3] and [4].

Holland began development of the LCS in 1971 [7]. Since then it has undergone modest experimentation. Some of the more recent work includes the application of the LCS to letter sequence prediction by Robertson and Riolo [8], multiplexer truth function learning by Wilson [9], predictive behavior learning by Carse [10], environment variable storage by Shu and Schaeffer [11], and learning by analogy by Zhou and Grefenstette [12]. Each of these tasks focused on the use of a single LCS. Dorigo and Schnepf, on the other hand, explored the use of multiple learning classifier systems in robotics, with each system controlling a different aspect of the robot. While they show that this control approach is effective, their setup does not explicitly pass messages between classifier systems [13]. The DLCS presented here goes beyond current learning classifier system research by introducing true networking into the LCS framework.

III. The DLCS

A. Overview

The DLCS extends the standard learning classifier system with the addition of a *network interface*, as shown in Figure 1. The figure indicates message flow with solid arrows and system control with dashed arrows. The execution cycle operates by reading messages from the environment through the input interface, selecting appropriate actions from the classifier list, and posting actions back to the environment via the output interface. With the DLCS, however, we extend this message passing to

the network. Now, messages can come from the input interface *and* the network interface and actions can get posted to the output interface *and* to the network. The network can also introduce new rules into the classifier list. We describe in detail these two cases, as well as a third message-passing case, in the next two sections. Since networks are asynchronous and tend to have a delivery delay that depends on external factors such as network congestion, available bandwidth, and transmission rate, the network interface contains *transmit* and *receive queues* for the purpose of buffering network messages so that the LCS does not have to wait for network response to continue execution. These queues are simply FIFO buffers which store outgoing and incoming messages.

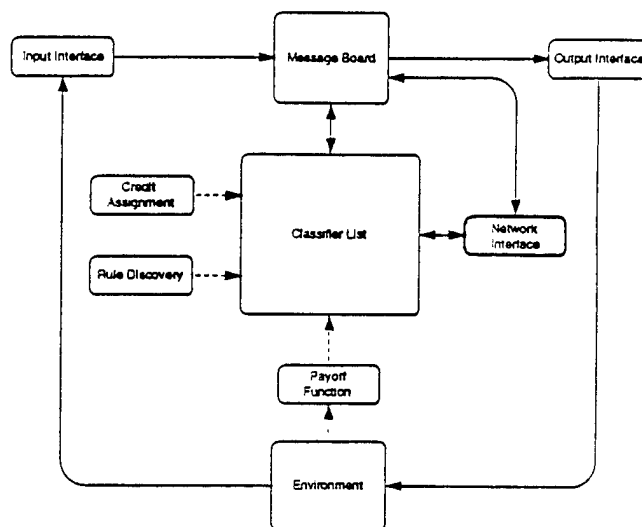


Figure 1. The DLCS

B. Network Message Types

In order to provide versatile and effective communication over the network, the DLCS paradigm provides three types of messages that can be passed over the network: classifier messages, action messages, and BBA strength adjustment messages. Classifier message passing occurs when an agent finds a useful rule and shares this rule with the other agents. Action message passing occurs when an agent periodically sends one or more of its selected actions to other agents in the network. Finally, since agents are passing action messages back and forth, rule chains can form between agents, and BBA strength adjustment messages must be sent between them.

Each type of message-passing has different implications for the DLCS. Classifier message passing allows one agent to share its learned knowledge with others. As will be shown in the next section, only classifiers with higher strengths can be passed, thereby ensuring that only "good" rules are shared between agents. Therefore, if one agent has learned part of a task more quickly than its fellow agents, that agent will share this learned information and help accelerate the learning process in the other agents.

Action message passing, on the other hand, provides a method for one agent to directly “talk” to the others. These received network action messages combined with environment interface messages work together to “fire” rules in the classifier list. The BBA payoff messages perform the same function as in the single-LCS case by encouraging chains to form between agents, thereby encouraging “discussion” among agents.

C. DLCS Execution Cycle

The following is the execution cycle for the distributed learning classifier system. The execution cycle is based on the standard LCS cycle, with items in *italics* representing DLCS additions.

1. From the input interface, read the messages from the environment and post them on the message board. *From the receive queue, read the action messages from other agents and post them on the message board. Also read the classifier messages from other agents and probabilistically replace weak rules in the classifier list with these new rules.*
2. Compare the messages on the message board with each classifier. Record a match for every classifier whose condition words have all been matched by these messages.
3. Calculate bids for each matching classifier. Probabilistically select classifiers to post.
4. Clear the message board and post the actions of the selected classifiers.
5. Send the messages on the message board to the output interface. *Send a subset of these messages and/or a fixed number of high-strength classifiers to the transmit queue.*
6. Adjust the strengths of classifiers. *Extract and process any BBA strength adjustment messages from the receive queue, and send BBA payoff messages to the transmit queue as necessary.*

Steps 1 and 5 of this execution cycle are responsible for reception and transmission of classifier and action network messages, respectively. We must establish rules for transmission and reception so that the number of messages sent over the network can be controlled and so that one can control the amount of influence agents have upon one another. We will discuss transmission first.

Transmission of action and classifier messages occurs on step 5 of the execution cycle. We state above that a “subset” of the selected actions should be sent. This subset is defined by two variables, the transmit bid threshold, B_{TX} , and the maximum number of actions to transmit, $N_{TX,action}$. The transmit bid threshold defines the minimum bid value required before an action is eligible to be sent over the network. By imposing a bid threshold, only those actions whose posting classifiers have a high strength will be sent, since the bid is based on the strength. $N_{TX,action}$

provides a way to control the amount of network traffic by imposing an upper limit on the number of actions that can be sent, in case all of the actions have large bids. If $N_{TX,action}$ is smaller than the number of actions eligible to be transmitted, then the actions with the largest bids are sent.

Transmission of classifiers on step 5 is also governed by two variables, the transmit strength threshold, S_{TX} , and the maximum number of classifiers to transmit, $N_{TX,classifier}$. The transmit strength threshold functions like the bid threshold in dictating a minimum strength required before a classifier can be transmitted. This threshold ensures that only “good” rules are sent over the network. $N_{TX,classifier}$ also helps control the amount of network traffic by limiting the number of classifiers that can be sent, since classifier lists are often rather large. These variables collectively influence the degree of coupling in the multi-agent application. Coupling should be “tight” enough that collective behaviors emerge, but not so tight that network bandwidth is threatened.

We should point out at this point that action and classifier messages are *broadcast* over the network to all other agents. As will be shown, each agent has the opportunity to discard received messages. Also, transmission of these two message types is globally paced by two time intervals: an action transmission interval, T_{action} and a classifier transmission interval $T_{classifier}$. If we define a unit of time as one iteration of the execution cycle, these intervals determine how many iterations occur between network transmissions. Larger intervals provide less inter-agent communication and therefore result in less coupling between agents. Smaller intervals provide more coupling.

Reception of action and classifier messages occurs on step 1 of the execution cycle. Since both action and classifier messages are broadcast, there will in general be more messages in the receive queue than were transmitted. Agents must have a method of filtering out the best messages from all those received. As with transmission, action and classifier message types are handled separately.

Action message reception is governed by two variables that are effectively the converse of the transmission variables, the receive bid threshold, B_{RX} , and the maximum number of actions to receive, $N_{RX,action}$. These variables are separate from their corresponding transmit variables because we may wish to put more stringent requirements on reception than transmission, since reception will have an impact on an agent’s behavior. In other words, an agent can be more liberal in its sharing of information, while putting a higher premium on the usefulness of received information. To achieve this end, one generally uses $B_{RX} > B_{TX}$ and $N_{RX,action} < N_{TX,action}$. Note that those received action messages that are not used are discarded.

Classifier message reception is also controlled by two variables, the receive strength threshold, S_{RX} , and the

maximum number of classifiers to receive, $N_{RX,classifier}$. These variables perform the same function as in action message reception. Again, we want the reception requirements to be more stringent because each rule that an agent accepts will replace a weaker rule in the classifier list. The more network-based rules accepted, the more agent rules replaced. While rule replacement is not a necessarily a detrimental occurrence, we want to ensure that only "bad" rules are replaced. Again, unused classifier messages are discarded. Note that there are no receive intervals for action or classifier reception since the frequency of reception is dependent on the frequency of transmission.

There are no restrictions on BBA payoff message transmission and reception, since this message passing is governed by the bucket brigade algorithm. The BBA dictates that *supplier* classifiers should be paid if they post an action message that fires one of the current classifiers. If these suppliers happen to be in another agent's classifier list, a BBA payoff message is sent over the network to that classifier. Note that in the BBA payoff message case, messages are not broadcast; they are sent directly to the agent getting paid. BBA transactions occur on execution cycle step 6.

Finally, depending on the task to which the DLCS paradigm is applied, one may wish to disable action passing or classifier passing. The DLCS has been designed so that action and classifier passing work independently, and therefore disabling one does not change the operation of the other. Also, disabling the bucket brigade algorithm disables BBA message passing.

IV. Multiple-Agent DLCS Example

As an example of the DLCS, we introduce the multiple-agent animat problem. An animat, or "artificial animal," is an extremely simple autonomous robot modeled after an animal [9]. The "animat problem" describes the autonomous robot's search for a goal in an obstacle-filled environment. We extend this animat problem to include multiple autonomous robots, each attempting to reach the same goal. We will show that the DLCS paradigm allows these animats to reach the goal faster and more efficiently than animats with standard LCS controllers.

We model a simple autonomous agent as a *treaded* robot subject to the *nonholonomic constraint*—the robot's velocity is limited by its position, thereby requiring that the robot travel only in the direction in which it is pointed. The robot is equipped with left and right goal and obstacle sensors, with the goal sensors having a much larger range than the obstacle sensors. However, the robot does not understand its sensor or kinematic systems; it must learn relationships between sensors and motion from credit assignment. Credit assignment is accomplished by evaluating an agent's action on each time step. The agent is rewarded for moving closer to the goal and being more

directly pointed at the goal. The agent is penalized for moving away and "looking away" from a goal and for crashing into an obstacle. The environment consists of an infinite plane with two long obstacles and a goal. The agents start behind both obstacles, such that the obstacles obstruct the most direct path to the goal.

The purpose of this simulation is to illustrate the effect of LCS network distribution. Therefore, we have disabled the Genetic Algorithm and the BBA in the standard LCS. For our DLCS settings, we use classifier message passing only, with $S_{TX} = S_{RX} = 80\%$ of the maximum strength on the classifier list at the time of transmission and reception, respectively. $N_{TX,Classifier} = N_{RX,Classifier} = 1$ and $T_{Classifier} = 1$. We use two agents in the animat scenario, and each agent's classifier list is initialized with *random* rules.

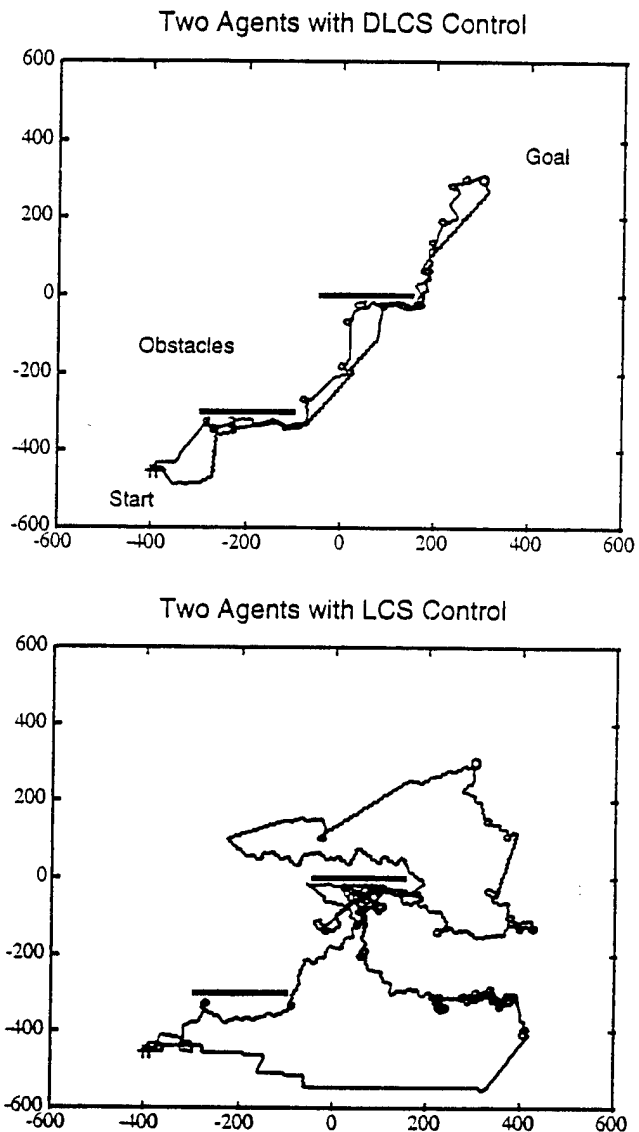


Figure 2. Simulation results comparing the DLCS with the standard LCS.

We have included the results of two simulations in Figure 2. In the first, the DLCS architecture is used for each agent. In the second, agents use the standard LCS architecture. In each case, the agents' classifier lists are initialized to the same identical set of random rules. As can be seen from the figure, the agents are much more successful in maneuvering around the obstacles and reaching the goal when using the DLCS paradigm than when they are given traditional LCS controllers with no inter-agent communication. Because classifier passing occurs on every iteration of each agent's execution cycle, agents' classifier lists tend to quickly converge to the most useful rule, in this case, a rule which moves the agents toward the goal. Without the DLCS architecture, agents are left to "fend for themselves," and their successfulness suffers accordingly.

V. Conclusion

While there are many facets of the standard learning classifier system that have yet to be fully explained, the LCS has proven to be useful in several areas of machine learning. To fully take advantage of some of the inherent characteristics of the LCS, we have introduced its distributed sibling, the DLCS. With the additions to the learning classifier system described in this paper, the DLCS can function as a paradigm for inter-agent communication and cooperation. We feel that the DLCS can be an effective tool in agent organization and coordination and can be useful over a wide range of distributed artificial intelligence tasks.

VI. References

- [1] Michalski, R.S., "Understanding the Nature of Learning: Issues and Research Directions," in *Machine Learning: An Artificial Intelligence Approach Vol. 2*, Boston: Kluwer Academic Publishers, 1987.
- [2] Rich, E., Knight, K., *Artificial Intelligence, second edition*, New York: McGraw-Hill, 1991.
- [3] Holland, J.H., *et al.*, *Induction: Processes of Inference, Learning, and Discovery*, Cambridge: MIT Press, 1986.
- [4] Booker, L.B., Goldberg, D.E., Holland, J.H., "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, vol. 40, nos. 1-3, September 1989, pp. 235-282.
- [5] Belew, R.K., Forrest, S., "Learning and Programming in Classifier Systems" in *Machine Learning Vol. 3*, Boston: Kluwer Academic Publishers, 1988.
- [6] Huhns, M.N., *Distributed Artificial Intelligence*, California: Morgan Kaufmann, 1987.
- [7] Wilson, S.W., Goldberg, D.E., "A Critical Review of Classifier Systems," *Proceedings of the 3rd International Conference on Genetic Algorithms*, June 1989, pp. 244-255.
- [8] Robertson, G.G., Riolo, R.L., "A Tale of Two Classifier Systems" in *Machine Learning Vol. 3*, Boston: Kluwer Academic Publishers, 1988.
- [9] Wilson, S.W., "Classifier Systems and the Animat Problem" in *Machine Learning Vol. 2*, Boston: Kluwer Academic Publishers, 1987.
- [10] Carse, B., "Learning Anticipatory Behaviour Using a Delayed Action Classifier System," AISB Workshop on Evolutionary Computing, Leeds, England, April 1994.
- [11] Shu, L., Schaeffer, J., "VCS: Variable Classifier Systems," *Proceedings of the 3rd International Conference on Genetic Algorithms*, June 1989, pp. 334-339.
- [12] Zhou, H.H., Grefenstette, J.J., "Learning by Analogy in Genetic Classifier Systems," *Proceedings of the 3rd International Conference on Genetic Algorithms*, June 1989, pp. 291-297.
- [13] Dorigo, M., Schnepf, U., "Genetics-Based Machine Learning and Behavior-Based Robotics: A New Synthesis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 1, January/February 1993, pp. 141-153.

Task Decomposition and Dynamic Policy Merging in the Distributed Q-learning Classifier System

Kevin L. Chapman and John S. Bay

Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0111

Abstract

A distributed reinforcement learning system is designed and implemented on a mobile robot for the study of complex task decomposition and dynamic policy merging in real robot learning environments. The Distributed Q-learning Classifier System (DQLCS) is evolved from the standard LCS proposed by J.H. Holland. We address two of the limitations of the LCS through the use of Q-learning as the apportionment of credit component and a distributed learning architecture to facilitate complex task decomposition. The Q-learning update equation is derived and its advantages over the complex bucket brigade algorithm (BBA) are discussed. Holistic and monolithic shaping approaches are used to distribute reward among the learning modules of the DQLCS and allow dynamic policy merging in a variety of real robot learning experiments.

1. Introduction

Recently, considerable interest has arisen in *robot learning*. The general theme in robot learning is that an intelligent machine is one that can sense its environment, learn how to cause change in its environment to achieve a goal, form plans to carry out tasks, and react to unpredicted external stimuli. In *unsupervised learning* the robot is given the ability to explore its environment in a trial-and-error fashion to collect data. From an evaluation of this data, the robot must learn a mapping from its input sensor values to its output effector actions. *Reinforcement learning* involves the use of feedback to reason about the quality of the robot's condition-action rules.

We have selected one unsupervised reinforcement learning algorithm, Holland's *Learning Classifier System* (LCS) [3,6,7], for study and implementation. The LCS is a rule based, message passing machine-learning paradigm that incorporates *planning* and *rule discovery* for intelligent problem solving in a dynamic environment. While the system we implement resembles

Holland's original LCS in structure, several additions and substitutions are included. After discussing the limitations the *bucket brigade algorithm*, the *apportionment of credit* (AOC) mechanism in Holland's LCS, we offer Christopher Watkins's *Q-learning* algorithm [10] as a replacement. We also examine an enhancement of the original LCS, the *Distributed Learning Classifier System* (DLCS) [1, 5]. The system we implement has the distributed capabilities of the DLCS, but it uses Watkins's Q-Learning mechanism for credit apportionment. We call this system the *Distributed Q-learning Classifier System* (DQLCS).

Monolithic systems suffer from slow learning due to the large size of the state space created by complex or multiple goal tasks [11]. This explosion in state space size is called the *curse of dimensionality*. *Task decomposition* [11] is a solution to the problem of complex task learning in which the overall task is divided into smaller pieces. Each individual task is given a control module whose objective is to learn only to achieve that task. Then, instead of learning over a single state space whose size is exponential in the number of tasks, the modular system learns over a linear number of constant sized state spaces. The capability of a system to then learn to coordinate these multiple behaviors, or *policies*, is called *dynamic policy merging* [11]. Many questions still remain about various *shaping* techniques, the schemes for combining reinforcement learning and distributed control on the same system. From our experiments, we comment on the effectiveness of the DQLCS at decomposing and solving robot learning problems using two distributed reinforcement techniques, *holistic shaping* and *modular shaping*.

2. Developing the DQLCS

The structure of the Learning Classifier System (LCS) is shown in Figure 1. Boxes represent the various components of the LCS, and arrows represent the flow of data through the system. The operation of the system is based on the use of lists of evolved production rules or

1997 IEEE Int'l. Symposium on Computational
Intelligence in Robotics, July 1997, pp 166-171

classifiers. Classifiers are *condition-action* pairs. Each classifier defines a possible state of the environment. Associated with each classifier is a *strength* value. The strength of a classifier is related to its current usefulness to the system as compared to all of the other classifiers in the system. The goal of the LCS is to adjust the strength of all of the classifiers over time until the classifiers most useful in achieving the desired goal are distinguishable from the rest. A *credit assignment* scheme is used to update the strengths of the classifiers. Credit assignment is recognized as the key to the success of the LCS.

The *input interface* and *output interface* are used for interaction with the environment. The input interface receives as input sensor values and encodes them into a message for posting on the message board. The output interface decodes the action part of the winning rule and sends the command(s) to the machine for execution. The *message board* is a bulletin board upon which messages describing the current state of the system are "posted". The *classifier list* is the rule base for the LCS that contains all of the system's rules and their associated strengths. *Credit assignment* or apportionment of credit (AOC) is used to adjust the strengths of the classifiers based on the positive or negative effects on the state of the system resulting from their use.

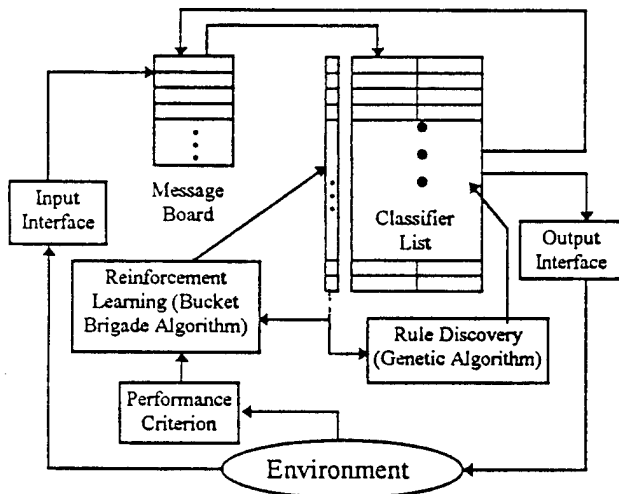


Figure 1. The Learning Classifier System (LCS) structure.

2.1. Q-learning vs. The BBA

The apportionment of credit component in the LCS architecture as originally proposed by Holland is the bucket brigade algorithm (BBA). In theory, the BBA solves the credit assignment problem by encouraging the formation of rule chains. Research has shown that the BBA in its original form could sustain pre-existing rule chains, but that it is not strong enough to successfully encourage chaining from a rule list consisting of initially

random strings [2, 5, 9]. The backwards propagation of rewards from the rules receiving the rewards to their supporters was observed to take many time steps. This slow learning process results in the need for a large number of initially long-running experiments before the first rules in the chain receive any reward.

Because of the time requirements imposed by real learning robot problems, we decided that the BBA is not sufficient for use in our system and that some other apportionment of credit algorithm is necessary. In the Q-learning classifier system (QLCS) [1] the BBA is replaced by Q-learning [10]. In a Q-learning system, each state-action rule has an associated *Q-value*. The *Q-value* is an estimate of the minimum cost-to-go associated with taking the rule's action when the state of the environment matches the rule's condition. After a rule is executed, the environmental feedback is used to update the its *Q-value*. Ultimately, the system reaches an *optimum policy*, a path of least total cost.

2.2 Derivation of the Q-learning Update Equation

Q-learning originated as a recursive algorithm for solving Markov decision problems. Markov processes contain a finite number of states, with each state having a finite number of possible actions. The probability of making a transition from one state to another is a function of the current state and not on any past history of the system. We call the set of n finite states in a Markov decision process $S = \{s_i\}$, $i=1, 2, \dots, n_s$. At each state, there is a set of n_{as} possible actions $A_s = \{a_{sj}\}$, $j=1, 2, \dots, n_{as}$. The probability of making a transition from state s_i to state s_j given the action a is $pr(s_i \rightarrow s_j) = p_{ij}(a)$. When an action $a(t) \in A_s$ is taken from state $s(t) \in S$ at time step t , the new state function $S(t+1) = \mu(s(t), a(t))$ determines the resulting state. Each state-action rule has an associated value, $c_i(a)$, that represents the instantaneous cost incurred by taking action a in state s_i . This probabilistic value is assumed to be either always positive or always negative and is estimated by the expected value: $E[c_i(a)] = \bar{c}_i(a)$.

We now establish the value function $V_\mu(i)$, the expected sum of all future discounted costs where the system starts at state s_i and follows the state function μ :

$$V_\mu(i) = \lim_{n_t \rightarrow \infty} E \left[\sum_{t=0}^{n_t-1} \gamma^t c_{s_t}(\mu(s_t)) \mid s_0 = s_i \right] \quad (1)$$

The summation includes all future states of the system, where s_t is the state of the system at time t and γ is the *discount factor*, $\gamma \in [0, 1]$. By applying the discount factor, we emphasize more immediate future costs over distant future costs. The inclusion of this

discount factor and the previously mentioned restriction on cost function sign facilitate the convergence of the summation [10].

Bellman's *principle of optimality* [8] is used to find the *optimal policy*, a policy that minimizes the future expected cost. Bellman's equation can be stated as the following:

$$V^*(s_i) = \min_{a \in A(i)} \left\{ \bar{c}_i(a) + \gamma \sum_{s_j \in S} p_{ij}(a) V^*(s_j) \right\}. \quad (2)$$

This equation says that the minimum total cost from the current state to the goal is the sum of the minimum of the expected instantaneous costs for actions from the current state and the minimum cost of going to the goal from the resulting next state.

By reformulating (2) as a recurrence relation, the costs-to-go may be estimated over repeated trials. Value iteration, a recursive estimation equation of the form

$$V^{(k+1)}(s_i) = \min_{a \in A(i)} \left\{ \bar{c}_i(a) + \gamma \sum_{s_j \in S} p_{ij}(a) V^k(s_j) \right\}, \quad (3)$$

has been shown to converge to the optimal value policy $V^*(s_i)$ for a given initial estimate $V^0(s_i)$ [10]. That is, if at the k^{th} iteration $V^k(s_i)$ is estimated as $V^k(s_i)$, then $V^{(k+1)}(s_i) \rightarrow V^*(s_i)$ as $k \rightarrow \infty$.

Watkins [10] reformulated Bellman's equation by adding Q-value notation:

$$Q^*(s_i, a) = \bar{c}_i(a) + \gamma \sum_{s_j \in S} p_{ij}(a) V^*(s_j). \quad (4)$$

In this equation, $Q^*(s_i, a)$ is the Q-value associated with taking action a from state s_i . Equation (5) shows the simplification of Bellman's equation (2) resulting from the substitution of Watkins's Q-value notation:

$$V^*(s_i) = \min_{a \in A(i)} Q^*(s_i, a). \quad (5)$$

Applying the value iteration technique to this simplified form of Bellman's equation gives us the following results:

$$V^k(s_i) = \min_{a \in A(i)} Q^k(s_i, a) \quad (6)$$

and

$$V^{k+1}(s_i) = \min_{a \in A(i)} Q^{k+1}(s_i, a). \quad (7)$$

Now the recurrence relation (3) becomes

$$\min_{a \in A(i)} Q^{(k+1)}(s_i, a) \leftarrow \min_{a \in A(i)} \left\{ \bar{c}_i(a) + \gamma \sum_{s_j \in S} p_{ij}(a) \min_{a \in A(j)} Q^k(s_j, a) \right\}. \quad (8)$$

Since this is known to converge to an optimal solution, we can rewrite (8) as

$$Q^{(k+1)}(s_i, a) \leftarrow \bar{c}_i(a) + \gamma \sum_{s_j \in S} p_{ij}(a) \min_{a \in A(j)} Q^k(s_j, a) \quad (9)$$

The recurrence relation of (9) provides an estimate for the Q-value of the state-action pair (s_i, a) in terms of an expected instantaneous cost and a weighted sum of minimum costs-to-go for the state action pairs (s_j, a) . We then approximate the unknown values $\bar{c}_i(a)$ and $p_{ij}(a)$ as:

$$\bar{c}_i(a) \approx c_i(a), \quad (10)$$

and

$$\sum_{s_j \in S} p_{ij}(a) V(s_j) \approx V(s_j). \quad (11)$$

In (10) and (11), the estimate of the expected instantaneous penalty is the single sample value of the incurred instantaneous penalty, and the expected minimum cost-to-go is estimated as the minimum of the estimated costs-to-go at the next state.

We can write the estimate of $Q(s_i, a)$ at the $(k+1)^{\text{st}}$ iteration by substituting (10) and (11) into (9):

$$Q^{(k+1)}(s_i, a) = c_i(a) + \gamma V^k(s_j). \quad (12)$$

Let $Q_t(s_i, a_t)$ be the current estimated minimum cost for executing action a in state s at time t . After taking this action, we update our estimate of $Q_t(s_i, a_t)$ using (12). The old and new estimates are combined in the relaxed Q-learning update equation:

$$Q_{t+1}(s_t, a_t) \leftarrow [1 - \alpha(s_t, a_t)] Q_t(s_t, a_t) + \alpha(s_t, a_t) [c_s(a_t) + \gamma V_t(s_{t+1})] \quad (13).$$

In (13), $\alpha(s_t, a_t)$ is a learning rate between 0 and 1. The learning rate thus provides a means to weight the combination of the Q-value's past estimation and the new measurement.

2.3. The QLCS Execution Cycle

After replacing the BBA with Q-learning and rule strengths with Q-values, the QLCS execution cycle is as follows.

From an initialized set of Q-values, Q_0 :

1. Observe the current state s of the system.
2. Compile a list of all eligible classifiers $E(s, t)$
3. Select a winning classifier using a stochastic selection method.
4. Pass the action part a of the winning classifier to the output interface to be executed.
5. Advance the system clock: $t = t + 1$.
6. Receive an immediate cost $c(s, a)$ for executing action a in state s at time t .
7. Examine the new message board.
8. Compute the new eligibility set $E(t)$ given the new environment state.

9. Rank all of the classifiers in $E(t)$ based on their Q-values.
10. Update the Q-value of the classifier chosen during the previous clock tick using the Q-learning update equation (13).
11. Make a probabilistic selection of the classifier with the maximum (or minimum) Q-value.
12. Goto 4.

In many problems the state space representation of the environment is too large for a monolithic learning system to explore in a feasible time period. A distributed architecture more suited to this type of problem was introduced by Dorigo [4]. His architecture allows for the distribution of *internal* LCSs for the study of learning problems involving real robots. The *Division of Labor* (DOL) architecture is proposed by Bay and Stanhope [1]. In the DOL learning system, separate modules focus on solving independent parts of a learning problem. This property closely matches the idea of task decomposition.

The DOL architecture contains a layer of *thinker* LCSs, each focusing on learning a specific behavior. Above this layer resides a *combiner* LCS. The job of the combiner is to coordinate the decisions of the thinkers and to choose the ultimate output action. This architecture also includes a *mediator* for distributing input sensor data and reward to the thinker LCSs. The decomposition of the input messages is the most important property of the DOL architecture. It greatly reduces the size of the state space search that must be completed in a learning application.

Besides the standard AOC problem, distributed systems have the problem of *shaping*, the decomposition and distribution of reward to each LCS in the system. Holistic shaping treats the entire learning system as a black box. The same reinforcement is blindly given to *all* of its LCSs. This reward scheme introduces some ambiguity problems to the system, however. Thinker LCSs may be rewarded for bad decisions or may go unrewarded for correct decisions. In modular shaping, the thinker LCSs are first trained independently. After they have obtained a suitable level of performance, their learning is "turned off", turning them into reactive systems. The combiner learns to coordinate the action messages of the thinker reactive systems to achieve the complex goal in a separate learning exercise.

3. Experimental Results

We observed the performance of various configurations of the QLCS and DQLCS when applied to typical real robot learning problems. The small mobile robot was equipped ultrasonic range finders and infrared sensors that detected modulated light from specially-

constructed "beacon" emitters. The problem environment selected for the learning system is shown in Figure 2. The problem is common in robotics: find the best path through an environment to a goal position.

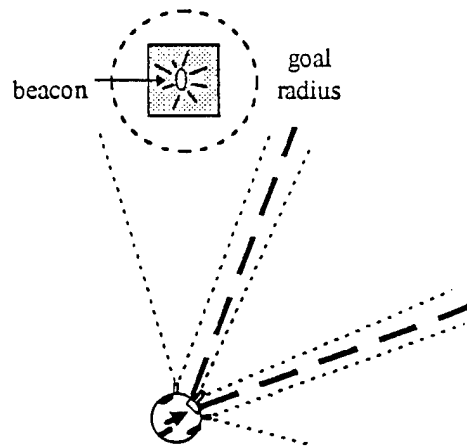


Figure 2. The goal seeking problem environment.

We used three infrared goal beacon detectors and a middle ultrasonic sensor as input sensors. For these sensors, the "docked", or goal, position was then defined as any position where the robot was "seeing" the goal beacon with *at least* the middle goal detector, and the range from the ultrasonic sensor was one unit of measure. Table 1 provides a breakdown of sensors and their associated penalty values used for instantaneous cost calculations in these experiments.

Sensor	Associated Penalty
Middle ultrasonic	$10 \times (3\text{-bit range value})$
Left Goal #2	$0.25 \times 50 = 12.5$
Middle Goal #2	50
Right Goal #2	$0.25 \times 50 = 12.5$

Table 1. QLCS penalty values for experiments

An upper limit of 50 time steps was used for each experiment. Regardless of the position of the robot, the system terminated the trial after the 50th clock cycle.

3.1 The Monolithic Approach

The docking problem was first studied using a monolithic Q-learning classifier system. The system used 6 bits of input: 1 bit from each of three goal beacon detectors and 3 bits from the ranging sensor. The output action string for each classifier was four bits: 2-bits for rotation and 2-bits for translation. The state space of the problem was then covered by 2^{10} or 1024 classifiers. The initial state of the environment is shown in Figure 2.

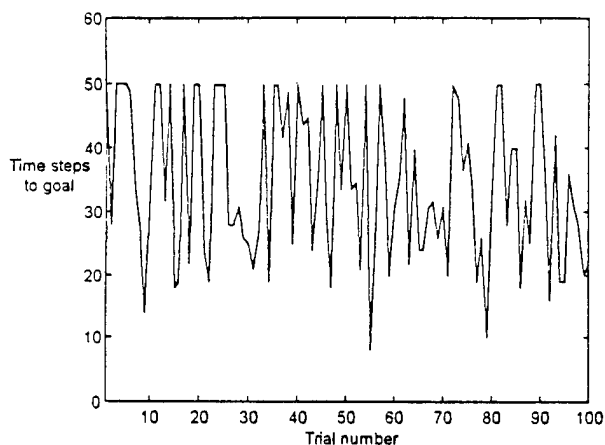


Figure 3. Learning curve for monolithic QLCS.

Figure 3 shows the monolithic QLCS's performance. This system was observed for 100 trials. For most of the experiment, it is difficult to find any system improvement. Within the final 15 trials, however, the system appears to be settling to some standard performance. It also "timed out" less as the experiment progressed.

3.2 The Distributed Approach

Next, we studied the same problem with the distributed architecture. Inside the DOL framework, we observed the performances of holistic and modular shaping. Table 2 shows the DQLCS configuration.

QLCS	Input bits/Source	Output	Clfs
Thinker #1	3/range	2-bit	32
Thinker #2	3/beacon detectors	2-bit	32
Combiner	2/output from Thinker 1 2/output from Thinker 2	4-bit	256

Table 2. DQLCS classifier configuration.

3.2.1 Holistic Shaping

Figure 4 shows the learning curve for the holistic system. There was only marginal improvement shown by this system. The system "timed out" during most of the first 20 trials. When the goal was reached, more than 25 time steps were always necessary. As the experiment progressed, the robot was able to find the goal more often and more quickly, but even then performance was only marginal.

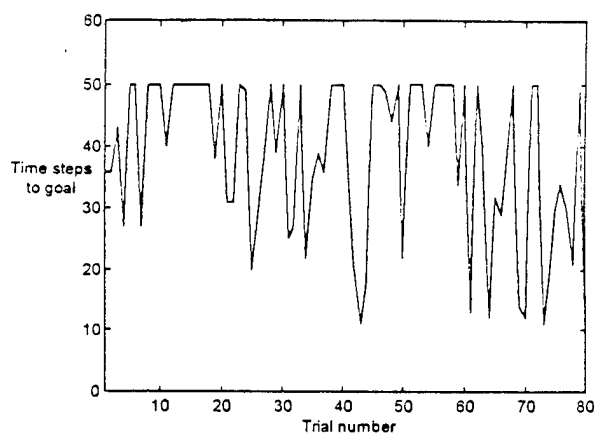


Figure 4. Learning curve for holistic shaping DQLCS.

3.2.2 Modular Shaping

For use with a modular shaping DQLCS, we divided the "goal-seeking" behavior into the sub-tasks of "approach goal" and "locate goal. The "approach goal" behavior involved translation only, while the "locate goal" behavior rotation only used rotation actions.

For the "approach goal" behavior, the QLCS used 3-bit range data as input. Four possible translation commands were selected by the 2-bit output action strings. The cost function returned a constant multiple of the range to the goal. The learning curve for the system is shown in Figure 5.

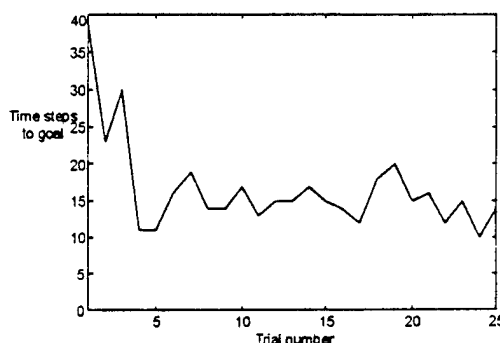


Figure 5. Learning curve for "approach goal" behavior.

The objective of the "locate goal" behavior was for the robot to learn to use rotation commands orient itself towards the goal beacon. The QLCS used a 3-bit input (one bit from each of the three goal detectors) and generated a 2-bit action string that was used to select from among four rotation commands. The goal detector penalties from Table 1 were used. To insure that the system learned the behavior completely, three different initial environments were used. For Trials 1-30, the starting position had the robot's left goal detector seeing the beacon. In Trials 31-50, the right beacon detector initially saw the beacon. In the remaining trials, the

robot was placed such that the beacon was directly behind it.

The results of the experiment are shown in Figure 6. This learning curve is actually a combination of three separate learning curves, one for each new starting position. In each of the three starting positions, we see that the system quickly learns the appropriate course of action to quickly reach the goal state.

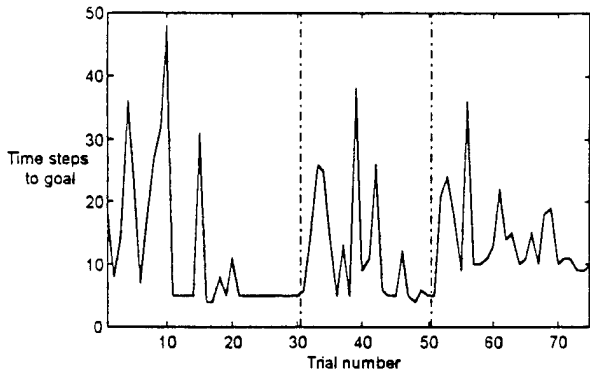


Figure 6. Learning curve for rotation behavior.

After the "locate goal" and "approach goal" behaviors were learned, the QLCSS' learning was "turned off". Using the two QLCSS as thinkers, an untrained combiner was given the "seek goal" task using modular shaping. Figure 7 shows the learning curve for the modular shaping DOL system. The improvement of the system from the first trial to the last is easily visible. As is shown in Figure 7, the system learned a path that required approximately 11 time steps after only 30 trials.

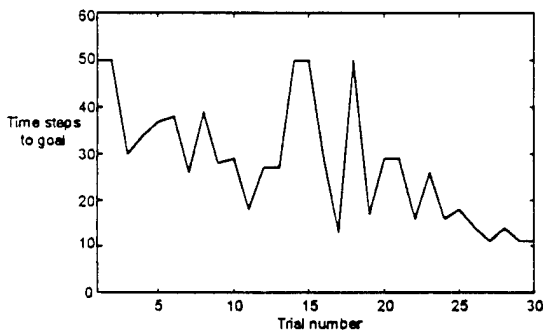


Figure 7. Learning Curve for modular shaping DQLCS.

4. Discussion of Results

From a comparison of the performances of the three classifier systems, we can see that the DQLCS with modular shaping was best suited for this problem. It appears that the size of the knowledge base in the monolithic system was its downfall. There simply were not enough visits to each state for the incremental learning algorithm to be successful. The ambiguous nature of the holistic shaping distributed system is its

major problem. It appears that reinforcement is assigned to the thinker classifiers in a manner that is much too haphazard. The learning process is slow; therefore, overcoming misguided reinforcement often takes too long to be practical in real robot applications.

The performances of the QLCSS show that Q-learning is an acceptable alternative to the BBA as the apportionment of credit component of the LCS. The results from the DQLCS experiments indicate that the distributed architecture using modular shaping is more time efficient at solving tasks that require the learning of complex behaviors than either a monolithic architecture or a distributed architecture using holistic shaping. While the DQLCS with modular shaping far outperformed the DQLCS with holistic shaping, it also required the infusion of much more domain specific knowledge. This requirement limits its application to a considerably smaller class of problems than may be attempted by the DQLCS with holistic shaping.

Acknowledgment

This work was supported by the Office of Naval Research under grant number N-00014-94-1-0676.

References

- [1] Bay, J.S., Stanhope, J.D., "Distributed Optimization of Tactical Actions by Mobile Intelligent Agents," *Journal of Robotic Systems, Special Issue on Mobile Robots*, 1996.
- [2] Belew, R.K., Forrest, S., "Learning and Programming in Classifier Systems," *Machine Learning Volume 3*, pp. 193-223, Netherlands: Kluwer Academic publishers, 1988.
- [3] Booker, L.B., Goldberg, D.E., Holland, J.H., "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, Vol. 40, no. 3, pp. 235-282, September 1989.
- [4] Dorigo, M., "ALECSYS and the AutoMouse: Learning to Control a Real Robot by Distributed Classifier Systems," *Machine Learning*, vol. 19, no. 3, pp. 209-240, 1995.
- [5] Gaff, D.G., *Architecture Design and Simulation for Distributed Learning Classifier Systems*, MS Thesis, Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, May 1995.
- [6] Holland, J.H., "A Mathematical Framework for Studying Learning in Classifier Systems," *Physica*, 1986, pp. 307-317.
- [7] Holland, J.H., *et al.*, *Induction: Processes of Inference, Learning, and Discovery*, Cambridge: MIT Press, 1986.
- [8] Larson, R.E., Casti, J.L., *Principles of Dynamic Programming, Part 1*, New York: Marcel Dekker, Inc., 1978, pp. 54-61.
- [9] Singh, S.P., "Transfer of learning Across Compositions of Sequential Tasks," *Proceedings of the 8th International Workshop on Machine Learning*, L. Birnbaum and G. Collins, eds, 1991.
- [10] Watkins, C.J.C.H., Dayan, P., "Q-Learning," *Machine Learning*, vol. 8, 1992, pp. 55-68.
- [11] Whitehead, S., Karlsson, J., Tenenbarg, J., "Learning Multiple Goal Behavior Via Task Decomposition and Dynamic Policy Merging," *Robot Learning*, Boston: Kluwer Academic Publishers, 1993.

Learning Classifier Systems for Single and Multiple Mobile Robots in Unstructured Environments

John S. Bay

Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061-0111
bay@vt.edu

ABSTRACT

The Learning Classifier System (LCS) is a learning production system that generates behavioral rules via an underlying discovery mechanism. The LCS architecture operates similarly to a blackboard architecture; *i.e.*, by posted-message communications. But in the LCS, the message board is wiped clean at every time interval, thereby requiring no persistent shared resource.

In this paper, we adapt the LCS to the problem of mobile robot navigation in completely unstructured environments. We consider the model of the robot itself, including its sensor and actuator structures, to be part of this environment, in addition to the world-model that includes a goal and obstacles at unknown locations. This requires a robot to learn its own I/O characteristics in addition to solving its navigation problem, but results in a learning controller that is equally applicable, unaltered, in robots with a wide variety of kinematic structures and sensing capabilities. We show the effectiveness of this LCS-based controller through both simulation and experimental trials with a small robot.

We then propose a new architecture, the Distributed Learning Classifier System (DLCS), which generalizes the message-passing behavior of the LCS from internal messages within a single agent to broadcast messages among multiple agents. This communications mode requires little bandwidth and is easily implemented with inexpensive, off-the-shelf hardware. The DLCS is shown to have potential application as a learning controller for multiple intelligent agents.

Keywords: machine learning, learning classifier systems, distributed artificial intelligence, production systems, distributed mobile robots, animat, genetic algorithms.

1. INTRODUCTION

As mobile robots become increasingly common for service and inspection tasks, the debate over the necessary level of complexity of their controllers continues. At one extreme is the purely reactive or behavioral approach, which espouses stimulus/response pairs over deliberative planning. At the other extreme are planning controllers, including hierarchical intelligent controllers, which may predict future states and generate control actions that consider a world model. Each approach has its advantages and disadvantages.

Reactive controllers are fast, simple, and are based on *a priori* assumptions and perhaps expert rules. They are primarily sensor-based and have no need for a world model because their actions are taken in response to real-time sensor inputs only. This economizes on memory requirements, temporal data processing, and deliberative computations. It also makes them quick to adapt to changing environments, since the only environment they recognize is what currently appears at their sensor ports. On the other hand, they tend to repeat past mistakes and have no ability to take advantage of even the simplest learning techniques. It is difficult for them to perform goal-oriented actions in complicated environments. Furthermore, it is unclear whether an algorithmic design procedure is possible for the creation of reactive rule bases, because their actions are modulated by the current state of the environment and other agents.

Mobile Robots X, Phila., PA, Nov 1995

Planning systems can better choose behaviors geared toward long-term rewards. They have long-term memory that can be used to represent their environment. They can process alternatives with symbolic manipulation and arrive at correct decisions that may be impossible with fixed rules and local senses. They can anticipate their consequences, evaluate their progress, and plan future actions. However their goal-motivation may make them situation-insensitive. Sensor input may be devalued in favor of their world models, and they therefore may be slow in adapting to a changing environment. Their world model itself may be taxing to acquire, maintain, and reconcile with contingencies.

A reasonable compromise for a practical control system is some sort of adaptive combination of the two approaches: a controller that follows stimulus/response rules at low levels of competence, such as in an execution layer, but which still allows the potential for planning, learning, and prediction. The reactive behaviors should be learned and evaluated on-line. Such a capability is potentially embodied in the learning classifier system (LCS) [3, 6].

An LCS is a computing machine that uses the architecture of a rule-based system, but also allows for memory and learning. It allows a user to parametrically tune it to weight its reactive vs. planning characteristics, thereby making it more or less goal-oriented or situation-oriented. Furthermore, the LCS functions as a modular, message-passing controller, so that data transmission is encapsulated in low bandwidth signals that simplify communications. In an intelligent agent, this characteristic allows us to construct behavioral modules that pass discrete messages with consistent formats. In this way, the LCS functions much like an object-oriented program (OOP), and indeed, OOPs are a favored simulation tool for their study. This feature also makes the LCS potentially extensible into multiple-agent domains.

In this paper, we will review the structure and operation of the learning classifier system and apply it to the learning control of a simple mobile robot performing as an animat. This controller will have to learn to interpret sensor data, drive the robot purposefully toward a goal, and avoid obstacles en route. We show that the learning properties of the classifier system enable us to build fast, sparse stimulus/response rule bases with default hierarchies. We also propose a distributed architecture for populations of such robots in message-passing environments. Simulation studies show that the LCS can provide adaptive controllers that are in some ways simpler than even deterministic Boolean functions.

1.2 Past Work

The need for learning techniques for the generation of reactive behaviors has been widely recognized recently. Among these, Kube [8, 9], used an adaptive logic network to learn the structure of a combinational logic circuit that controls the movement of multiple mobile robots. Ram [11] and Sims [14] both used genetic algorithms to learn reactive behaviors, with [14] taking the additional step of combining the structure of the robot into the evolutionary process along with the controller. An advantage of using the genetic algorithm is that the system remains continually adaptive and can adjust through incremental changes in structure and function. It also allows the designer to impose arbitrary expert rules on the system as constraints or defaults, and to preserve them under evolutionary operators through elitism [10].

Learning classifiers themselves have been used to control the behavior of animats, most directly by Wilson [16], and Dorigo [4]. Wilson proposes the environmental reward technique that we adapt for this paper, but shows experiments for the learning of a logic function only (the multiplexer problem). Dorigo uses the classifier system to track a moving target, and shows some behaviors consistent with predictive abilities in the learned behavior.

Our approach is closest to that of Husbands [7] and Wilson [16], as we apply the LCS to an animat model in a static environment. Our main contribution here is to show evidence that in such a situation, reactive learning is effective without planning abilities, and that a fast learning algorithm can economize on even deterministic logic. We also demonstrate that the transient-message-passing nature of the LCS extends directly to multiple agents, so that the same control architecture might be used in arbitrarily sized populations.

1.3 Review Of Learning Classifier Systems

The learning classifier system was developed by Holland and his associates after the effectiveness of the genetic algorithm had been established [3, 6]. The LCS combines genetic operators and apportionment of credit schemes together with an architecture

that facilitates implementations in physical devices such as robots equipped with sensors, actuators, and processors. Given below is our formalism for the LCS.

The overall arrangement of the LCS is shown in Figure 1. It consists of a set $C = \{C_i\}$, $i = 1, \dots, n$ of classifiers each with k ($k \geq 1$) conjunctive condition words c , and a single action word a . At an iteration time t , each classifier C_i is therefore represented by a $(k+1)$ -tuple of words, $C_i(t) = (c_{i1}(t), \dots, c_{ik}(t), a_i(t))$. Each of these words is a string of length l with elements taken from the set $\{0, 1, \#\}$.

Separate from this is the message board, which at all times consists of at most m messages; $M(t) = \{m_1(t), \dots, m_m(t)\}$. Each message is a single word of the same length, l , and consists of elements from the set $\{0, 1\}$.

A data path exists from the action words of the classifier list to the message board. Execution in its simplest form proceeds as each classifier compares its conditions to the message board. A condition c_{ij} is said to match a message m_p , denoted $c_{ij}(t) \approx m_p(t)$, if each 0 and 1 in c_{ij} has an equal bit in the same position in m_p , and the $\#$ -symbols act as don't cares. An eligibility index set E is created that contains a list of the classifiers C_i for which all conditions are matched by at least one message in M ; i.e., $E(t) = \{i : c_{ij}(t) \approx m_p(t) \text{ for all } j = 1, \dots, k \text{ and any } p\}$.

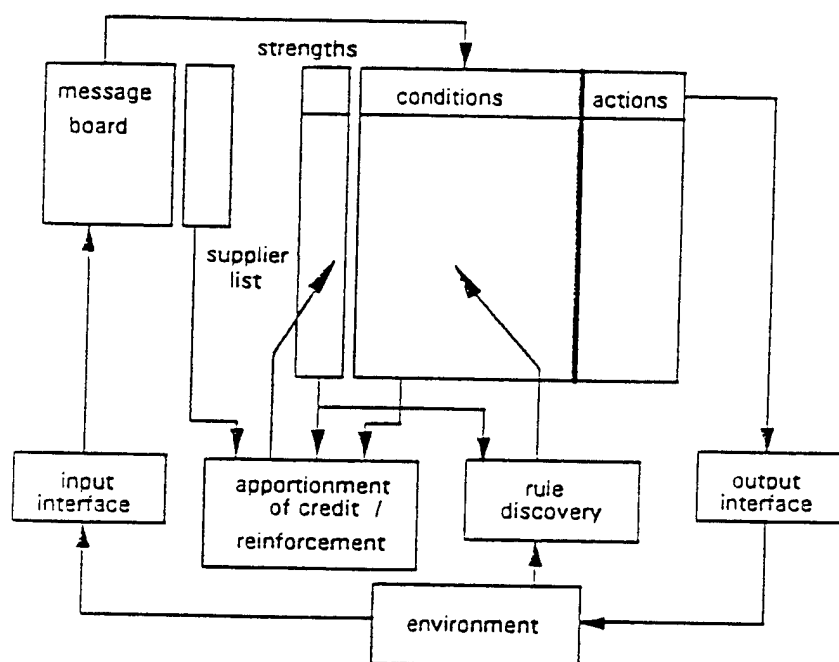


Figure 1: The architecture of the learning classifier system.

In conjunction with each rule is a *strength* value that signifies the overall merit of the rule over its lifetime. The use of the strength value varies with the implementation of the apportionment of credit and reinforcement components, but it can always be interpreted as a relative figure of merit.

Also necessary is a supplier list $L(t) = (L_1(t), \dots, L_m(t))$ of length m , which contains integers denoting the identity of the classifier that posted the corresponding message at the previous time step t : for each $j = 1, \dots, m$, $L_j(t) = i$, where $m_j(t) = a_i(t-1)$. These are known as the *suppliers* of those messages. This list will later allow a classifier to know which of its counterparts posted a message that now matches one or more of its conditions. In the case that message j was posted by the

sensor interface, a special symbol (such as 0) may be entered in L that conveys this information. This will enable us to create the sets of *supporters* that will be important for our apportionment of credit scheme.

It is generally desirable to have fewer messages than eligible classifiers; $m < E^*$, where E^* is the cardinality of the set $E(t)$. We therefore will require an arbitration mechanism that decides which of the classifiers in $E(t)$ is allowed to post its action on the message board. Known as roulette-wheel selection, classifiers are selected from $E(t)$ to post their actions probabilistically, with $\max(m, E^*)$ classifiers chosen according to a bid value associated with each classifier. Classifier i 's bid, $B_i(t)$, is a figure of merit that may encompass several factors, including its strength. In order to generate a new message board at time t , the roulette-wheel algorithm simply chooses elements of $E(t)$ where the probability of being selected is

$$p(i) = \frac{B_i(t)}{\sum_{j \in E(t)} B_j(t)} \quad (1)$$

or, using the Boltzmann distribution,

$$p(i) = \frac{e^{TB_i(t)}}{\sum_{j \in E(t)} e^{TB_j(t)}}$$

where the "temperature" parameter T allows us to control the likelihood of allowing low-bidding rules to post. This selection mechanism is sometimes referred to as the *conflict resolution* component of the classifier system.

The message board at the next time iteration consists of all sensor messages and the action parts of all classifiers so chosen. We will call this set of winning classifiers \hat{C} .

Messages may be thus posted by the classifiers themselves, or they may be derived from a sensor interface. Such an interface might interpret raw sensor data and set appropriate bits in a message. These messages generally are not subject to the roulette wheel competition with the actions of the classifiers, because sensor data may be necessarily accessible at any time. In practice, the number of sensors will dictate the minimum allowable length l for the messages*.

A counterpart to the sensor interface is the effector interface. This interface recognizes postings that have a true output tag (pre-specified bit position(s)), and routes them instead to a set of actuators. Optionally, the outputs may go to both the effector interface and the message board, in order that classifiers may see the current state of the actuators.

1.3 Apportionment of Credit and Reinforcement

The apportionment of credit block in the diagram can take a number of forms. As originally proposed by Holland, this was the so-called bucket-brigade algorithm (BBA). The BBA is a mechanism by which a classifier selected to post an action pays for this privilege by sharing its strength with those classifiers that previously posted messages that now enable it; i.e., its supporters, determined from the list of suppliers posted along with the message board. (The distinction between a supporter and a supplier is that every classifier that posts to the message board is a *supplier*, but only those that enable a winning classifier at the subsequent time are *supporters*.)

Further, the BBA provides that the actual bid be a scaled product of a classifier's strength, its *specificity*, and its *support*. The specificity is the fraction of *non-don't cares* in the rule, thereby weakening *default* rules. The support is the sum of the strengths of the classifier's supporters. This encourages rules that are triggered by other rules that are themselves relatively strong. Details of the BBA algorithm may be found in [2, 3, 6, and 13]. In principle, the BBA emulates a consumer/supplier economy

* An alternative to this designation that we have recently adopted is to concatenate each sensor input string with an internally posted message. This makes all messages a consistent format (part sensor / part internal message) and ensures that an external action may always be taken simultaneously with an internal message posting.

that encourages backward-chained sequences of rules, thus accomplishing goal-orientedness. The reinforcement component is primarily responsible for encouraging situation-orientedness.

The reinforcement component is closely tied to the environment, and may also use an intelligent tutor or critic. This component adjusts the strength value of each rule according to the immediate payoff as determined by the evaluation of actions as they are taken. For example, a reward schedule may be set up to highly reward progress toward a goal and penalize proximity to a hazard. More elaborate reinforcement schemes may also be designed that reward all rules *similar to* [16] a winning rule, or penalize rules that have not fired in a long time (via *taxes*). Because this reinforcement component strengthens rules for taking appropriate actions in each situation, it is most useful for encouraging situation-orientedness.

1.4 Discovery Component

The discovery component can take many forms also. Most often, it consists of a set of genetic or evolutionary operators that adjust and create rules using the strength as the "goodness of fit" value. We employ the most common of these, which are crossover and mutation. We also use elitism, which prevents good rules from being destroyed by the random evolutionary operators [10].

Each time a new rule is created by the evolutionary operators, an old one is replaced so that the size of the rule base remains constant. Selection of the weak rules for replacement is performed by the same roulette-wheel selection (equation (1)) as for the selection of eligible rules for action posting, except that reverse strengths are used.

2. THE LCS AND THE ANIMAT PROBLEM

The term *animat*, as used by Wilson [16], refers to an autonomous mobile agent that lives in an unstructured environment. Its goal is to travel from a starting point to a goal site while avoiding obstacles en route. While best imagined as a mobile robot, the concept effectively serves as a model for arbitrary problems in state spaces, and in particular, Markov decision processes.

For our purposes, we create the animat from a differentially steered mobile robot model. Figure 2 shows the model and the kinematics. We use realistic kinematics for the motion of the vehicle rather than the cell-stepping model in other works not only because it is more physically accurate, but because it requires no tessellation of the state space and may preserve the real dynamics of the robot. The robot has two actuators, one on each side of the vehicle, and four sensors. The sensors each have a detection range, shown in the figure to be larger for goal detection than obstacle detection. It is assumed that the robot has no knowledge of any inertial reference frame, so that all of its measurements are relative to itself. The motion of the robot is described by the set of equations:

$$\begin{aligned} V_{robot} &= \frac{1}{2}(V_{right} + V_{left}) \\ \dot{\theta} &= \frac{1}{2r}(V_{right} - V_{left}) \end{aligned} \quad (3)$$

with the constraint that V_{robot} is in the direction θ . Equations (3) may be discretely approximated to give estimated position p and orientations θ :

$$\begin{aligned} p_{robot}(t) &\equiv p_{robot}(t-1) + \frac{\tau}{2}(V_{right}(t-1) + V_{left}(t-1)) \\ \theta(t) &\equiv \theta(t-1) + \frac{\tau}{2r}(V_{right}(t-1) - V_{left}(t-1)) \end{aligned} \quad (4)$$

where τ is the size of the time-step. Note that the robot cannot measure its own position and orientation; only their derivatives. These equations are used for simulation purposes.

With this model for the robot, we will seek a rule-based controller that reads the sensor signals and produces proper (on/off) actuator outputs. This controller will not initially know even the kinematics of the robot; e.g., it will not know that turning on the left motor will cause it to turn right or that turning on both motors will cause it to go forward. This knowledge will have to

be learned. Furthermore, the robot will have to learn to navigate in its world of goal and obstacles. Feedback to enable both of these learning behaviors will come from the sensors. However, while the classifier system's input interface will tell the controller only binary information (object/no object) from the detection region, the environmental reward component will have analog data on the distance to the goal and from the nearest obstacle (provided they are within the detection region). Thus the reward can be based on proximity values. The rules fire in response to object *detectors*, while the reward utilizes object *range sensors*. Note that our sparse, low-resolution sensor data will result in *perceptual aliasing* [15], so that the robot will have difficulty distinguishing different world-states. This is intentional and, we believe, realistic.

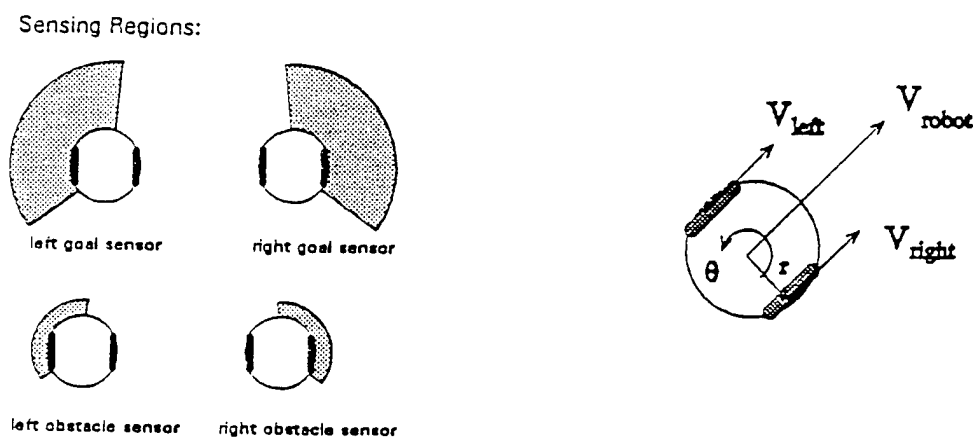


Figure 2: Sensor and kinematic models for the animat-robot. Modeled after ultrasonic and infrared sensors, respectively, the goal sensors have a much larger range than the obstacle detectors, and both sensor ranges overlap dead-ahead (so that the robot is not doomed to a life of tacking back and forth as it travels.)

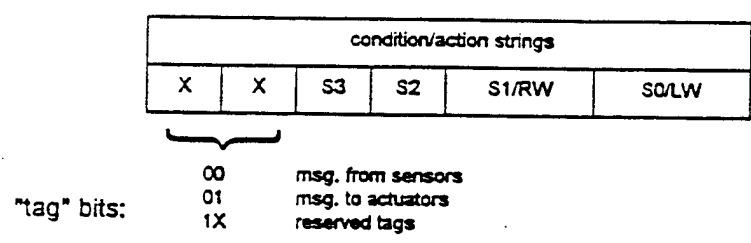
2.2 Application of the Deterministic Classifier System

As a preliminary baseline experiment, we first show how this problem can be solved with a fixed rule base that implements Boolean logic. This will give a deterministic classifier system that uses no bucket brigade. This hand-coded classifier system can be represented by the equations:

$$\begin{aligned} \text{left_wheel} &= s_0 + \bar{s}_2 s_3 + \bar{s}_1 \bar{s}_2 \\ \text{right_wheel} &= s_2 \bar{s}_3 + s_1 \bar{s}_2 + \bar{s}_0 \bar{s}_3 + \bar{s}_0 s_2 \end{aligned} \tag{5}$$

where s_0 , s_1 , s_2 , and s_3 denote, respectively, the right and left goal sensors, and the right and left obstacle sensors.

To implement these functions in the deterministic classifier, we will use $l = 6$, $k = 1$, $m = 1$, with the only message on the message board being reserved for the sensors, and all postings going directly to the output. The format for the condition/action strings are shown in Figure 3:



With this format, one can show that the system of the following seven classifiers in Table 1 successfully implements the Boolean functions (5):

conditions						actions (rightmost two bits posted to actuators (on/off))					
0	0	0	#	#	0	0	1	#	#	#	1
0	0	0	1	#	#	0	1	#	#	#	1
0	0	#	0	1	#	0	1	#	#	#	1
0	0	#	1	#	0	0	1	#	#	#	1
0	0	#	#	#	1	0	1	#	#	1	#
0	0	1	0	#	#	0	1	#	#	1	#
0	0	#	0	0	#	0	1	#	#	1	#

Table 1: The fixed, deterministic classifiers that implement the Boolean logic of equations (5). The first four classifiers represent the logic for the left wheel, and the last three give the right wheel.

When this fixed classifier controller is executed, the seven classifiers in Table 1 become the rules in a reactive controller. Applied to an animat problem with a single goal and two wall-shaped obstacles, we get the behavior depicted in Figure 4a. Note again that these rules are fixed and there is neither an environmental reward, bucket brigade, nor discovery component in effect. The purpose of this exercise is not only to demonstrate the performance of the classifier architecture, but to show an "optimal" animat behavior for future comparison.

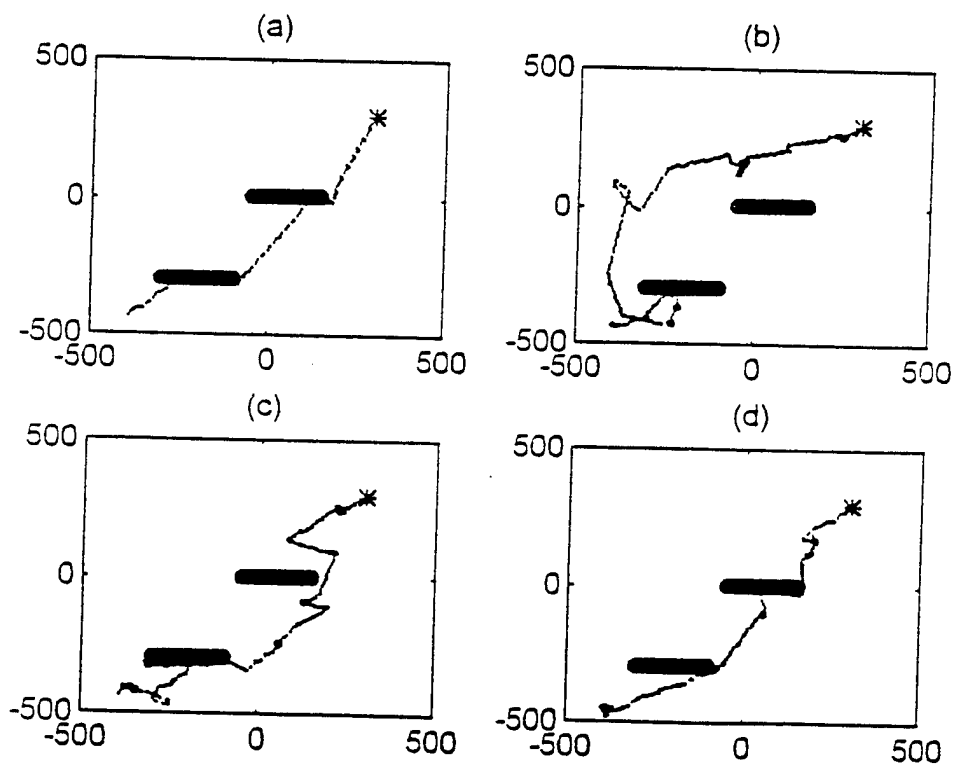


Figure 4: Four trial runs of the LCS (without bucket brigade). The first (a) is the "baseline" run executed with the deterministic classifier given in table 1. In each figure, the horizontal bars are obstacles, and the robot must start from the lower left position to the star at the upper right.

3. USING THE LEARNING CLASSIFIER

Given the performance of the deterministic classifier as seen in Figure 4a, one would expect that a learning classifier might arrive at a similar set of rules after experimentation. Although the deterministic control rules generated (5) were not a unique choice, we can follow the lead of past literature and consider them a "correct" set by which to measure the learned classifiers [13, 16]. The goal of this experiment is to learn to choose a suitable set of learning parameters, including the genetic operator probabilities and the environmental reward/penalty functions, and then analyze the rules that result. We might guess, incorrectly, that when the classifier system has learned to guide the animat to the goal, that we might "read" the classifiers in the list, translated by the word format of Figure 3, and derive Boolean logic similar to equations (5). A similar end result was produced with an "adaptive logic network" in [8, 9].

Because we are temporarily investigating the learning of a combinational logic function, we eliminate any sequential processing by turning off the bucket brigade. We will again use $l = 6$, $k = 1$, $m = 1$, and $n = 32$. We will initialize these 32 classifiers with random arrangements from $\{0, 1, \#\}$ (except for the fixed "tag" columns which remain invariant even under genetic operations). We use a (dimensionless) step size of $\tau = 1.5$ and a robot radius of $r = 1$.

When any classifier posts an action to the output interface (which happens in this case at every iteration, since there are no posts to the message board), a payoff value v is computed according to:

$$v = 5W_{goal}(3f_{goal} - 1) + 5W_{angle}(3f_{angle} - 1) - W_{crash}f_{crash}$$

where W_{goal} , W_{angle} , and W_{crash} are, respectively, chosen reward amounts for moving closer to the goal, pointing more directly at the goal, and crashing into an obstacle. Flags f_{goal} , f_{angle} , and f_{crash} are set to 1 if these three conditions are detected; 0 otherwise. The first two terms of the function above reflects a reward of W if a good condition is achieved, and a penalty of $0.5W$ if the result is bad. Crashing is always penalized.

Whenever an action is taken, a set A is created using the posting classifier(s) \hat{C} as follows:

$$A = E \cap \{i : a_i \approx a_w \text{ for any } C_w \in \hat{C}\}$$

Then we use the reward distribution function to adjust the strength S_i of each classifier:

$$S_i(t) = \begin{cases} S_i(t-1) + \frac{v}{|A|} & \text{if } i \in A \\ S_i(t-1) \cdot (1 - W_{pen}) & \text{if } i \notin A \end{cases}$$

where W_{pen} is a penalty factor that is (optionally) used to penalize classifiers that are in E , (their conditions match the sensors), but whose actions are not the same as the winners' actions. In the experiments given here, we use $W_{pen} = 0$, with the reasoning being that we prefer to encourage the emergence of diverse classifiers relevant to any input, rather than force classifiers to agree on an output in response to a particular input.

For the discovery component, we apply mutation and crossover every 50 iterations, so that the reward component has sufficient time to separate good rules from bad ones between genetic operations. To perform discovery, we select one parent C_1 based on

* The classifier is "sub-symbolic," meaning that it learns rules that are unconstrained by any syntax that the user is prepared to translate into English. One may look at an operational LCS and not be able to decipher its rules.

roulette-wheel selection considering the strengths of the classifiers. We then, with probability χ , select a second parent C_2 again using the roulette wheel. If a second parent is chosen, a crossover is performed between C_1 and C_2 at a random bit position. If no second parent is selected, the first parent is merely replicated. The offspring C_{new} is then provided an initial strength by sharing the strengths equally with its one or two parents. It is used to replace a weak classifier by roulette-wheel selection based on a fitness vector $\max_j(S_j(t)) - S_i(t)$. This prevents the strongest classifiers from being killed to make room for a mere mutation.

Mutation is then performed, bit-wise, on the entire set C , with probability μ . This mutation is three-way, such that a 0 might remain 0 (with probability $1 - \mu$), or with probability μ , become a 1 or # (50% chance of each).

4. SIMULATION RESULTS

Shown in Figures 4(b-d) are three trial runs of the learning classifier system controlling the animat to reach the goal. These are typical trial runs and show behavior similar to the hand-coded version (Figure 4(a)). In (b-d), all rules are initialized at random.

5. EXPERIMENTAL RESULTS

To test real-time performance and the effects of sensor and actuator errors, a learning classifier system was programmed to control Curly, a modified RWI B12 robot [12]. Curly is outfitted with a Motorola 68HC11 based microcontroller with 32K of external RAM for programming. It has a different set of sensors than the simulated model: First, it has three diffuse-reflective obstacle detectors with ranges of approx. 10cm, mounted at dead-center and approx. 30° to either side. It also has a long-range infrared detector that detects only modulated infrared from a special source. This source "beacon" represents the goal and emits IR at 40 KHz modulated by 160 Hz. Last, it has an ultrasonic range finder mounted so that it points in the same direction as the beacon detector. This sensor is used by the reinforcement component to determine progress toward the goal, but its range readings are not available to the LCS, thereby making the sensing resolution of Curly roughly equivalent to the simulated model. Figure 5 shows Curly's sensor apparatus.

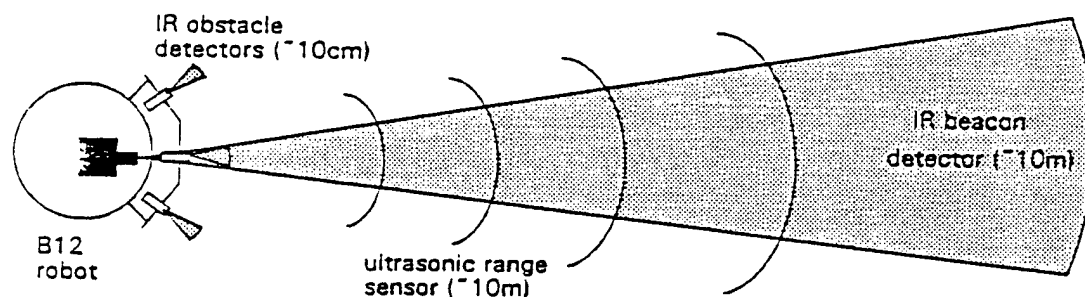


Figure 5. Curly, the modified B12, with sensor ranges.

The LCS is surprisingly simple to program on a real robot, and can be done with few lines of assembly or C code. Most of the extensive programming necessary for the simulation runs was in simulating the robot itself and its environment, as well as a user interface [5]. The actual mechanism of the classifier system (Figure 1) is quite efficient, with the genetic algorithm being probably the most time-consuming. We note that Curly is not a treaded vehicle, as is the simulated model, but is instead synchro-drive. The natural outputs are therefore not *right_wheel* and *left_wheel*, but *translate* and *rotate*. The controller, although it now writes to these two actuator behaviors, is unaltered. Sensorimotor behaviors are learned entirely from uninterpreted data, so the new kinematics are learned automatically.

With an LCS programmed on Curly, Figure 6 shows a representative search trial. Tabulated data over many trial runs is not presented here because the variance of the results (number of steps to reach the goal) is very high, and few trends, as yet, are apparent. Experiments are now underway to repeat the trials with an operational BBA and with Q-learning in the hopes that a learning curve will be more readily apparent.

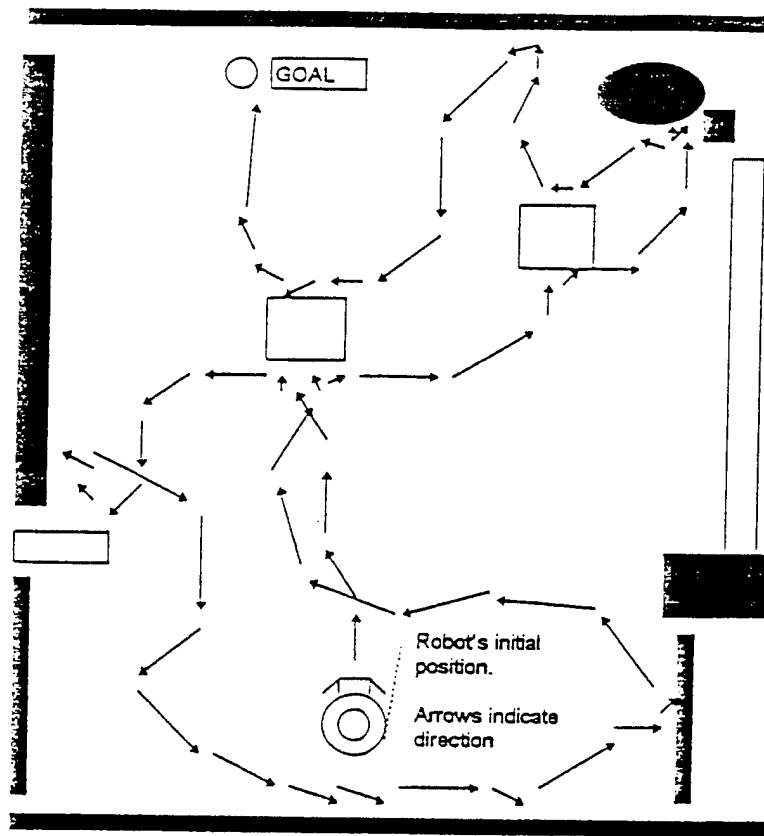


Figure 6. Trial run for a mobile robot in an unknown, unstructured environment. The environment is a cluttered laboratory with rectangular and ellipsoidal obstacles.

6. THE DISTRIBUTED LEARNING CLASSIFIER SYSTEM

A popular extension to reactive mobile robot controllers is the concept of a distributed network of mobile robots. With this in mind, we propose the distributed learning classifier system (DLCS). The DLCS is designed with the primary constraints that *i*) communications should be kept to an absolute minimum in order to avoid exacerbating a combinatorial explosion in overhead and routing requirements, and *ii*) communications should be limited to sporadic, self-contained messages of fixed format that may be addressed to any individual or simply broadcast to the population as a whole. It is widely noted that simple broadcast communications can be very useful in multi-robot coordination [1].

The architecture we propose is depicted in Figure 7 below. In the architecture, two extra connections are added to the separate learning classifier systems. First, a *transmit queue* is offered as a third option for output messages. That is, actions may be posted to the internal message board, to the output interface, or to the transmit queue, which places it on the communications medium. Second, a *receive queue* is added to the message board so that it can take data from the input interface or classifier list, as before, or from the communications bus.

If the assumed communications medium is perfect, this architecture may be viewed as a concatenation of each message board and each classifier list, since all postings to or from one are available to another. The DLCS would thus act as a single,

distributed classifier system. In reality, though, the communications channel is a protected resource, and it may not even be desirable for all classifiers to be so tightly coupled. An alternative would therefore be a rationing of the communications channel such that each classifier performs its duties as an individual, but selected (*i.e.*, particularly strong) classifiers could share actions with others through net broadcast. A second option, which implements the "learning by watching" paradigm for distributed learning [15] would be to allow entire rules (condition and action) to be shared, based on their relative strengths.

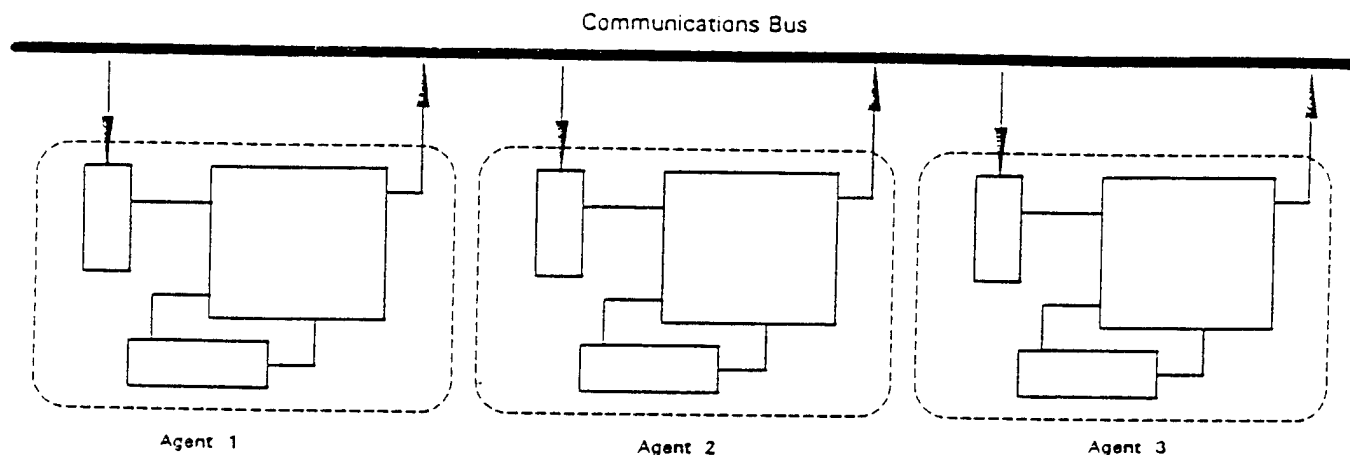


Figure 7. Architecture of the DLCS. Agents may selectively post to and read from the communications channel, implemented as a distributed message board.

Such a DLCS is currently under study and is being adapted to a number of distributed optimization problems as well. Simulations with multiple mobile robots are presented in [5].

7. CONCLUSIONS

Although it is clear that the LCS forms an appropriate architecture for the execution of reactive rules in mobile robots, real-world experiments are inconclusive as to its learning powers. Although the simulated behavior was quite successful, experiments with real robots shows a very weak learning curve. We note, however, that persistent learning is due, in part, to a successful apportionment of credit component, which we have not used in this study. Part of the reason for this was our observation, under simulated scenarios, that the BBA successfully maintains a sequence of classifiers whose actions are appropriately rewarded, but it is not very good at working in conjunction with the discovery component to *generate* viable sequences. There are some indications that evolution of *teams* of rules and specialized evolutionary operators may be more useful in this regard [2, 13].

Instead, an analysis of the strengths vs. time for individual rules (not shown here) indicates that rules live a relatively short life under the evolutionary operators. Our robot lives in a sensory-sparse environment and uses a fast rate of learning (high mutation probabilities of 0.1 – 0.3) to generate strong rules that "live for the moment." At any given time, only two or three rules, on average, have significant strengths, after which their usefulness wanes, they die and are quickly replaced. We therefore observe that we may use considerably fewer than the number of deterministic hand-coded Boolean rules (seven) needed in our example, as long as we trade them when they become irrelevant. The robots with this controller are extremely situation-oriented.

Future work is concentrating on the DLCS, as this is an entirely unexplored area for multiple agent coordination. The inherently message-oriented nature of the system translates directly to existing communications protocols, and promises speed-up through parallelism for computational optimization tasks in general.

8. ACKNOWLEDGMENTS

This work was supported in part by the Naval Research Laboratory under grant no N00014-93-1-G022 and in part by the Office of Naval Research under grant no. N00014-94-1-0676. The author would like to thank Doug Gaff for his assistance with simulations, and George Chrysanthakopoulos, Kevin Harrelson, Jason Thweatt, and Michael Spruill for their experimental work with Curly.

9. REFERENCES

- Altenburg, and M. Pavicic, "Initial Results of the Use of Inter-Robot Communication for a Multiple Mobile Robotic System," *IJCAI '93 Workshop on Dynamically Interacting Robots*, pp. 95 - 100, 1993.
- R. K. Belew and S. Forrest, "Learning and Programming in Classifier Systems," *Machine Learning 3*, pp. 193 - 223, 1988.
- L. B. Booker, et al, "Classifier Systems and Genetic Algorithms," *Artificial Intelligence*, vol. 40, pp. 235 - 282, 1989.
- M. Dorigo and U. Schnepf, "Genetics-Based Machine Learning and Behavior-Based Robotics: A New Synthesis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 1, pp. 141-153, Jan/Feb 1993.
- D. G. Gaff and J. S. Bay, "The Distributed Learning Classifier System," to appear, *LASTED-ISMM International Conference on Parallel and Distributed Computing*, October 1995.
- H. Holland, et al, *Induction: Processes of Inferences, Learning, and Discovery*, Cambridge: MIT Press, 1986.
- Husbands and I. Harvey, "Evolution versus Design: Controlling Autonomous Robots," *Third Annual Conference on AI, Simulation, and Planning in High Autonomy Systems*, Perth, Australia, pp. 136 - 146, July 1992.
- R. Kube and H. Zhang, "Collective Robotics: From Social Insects to Robots," *Adaptive Behavior 2*(2), Cambridge: MIT press, 1993.
- R. Kube, et al, "Controlling Collective Tasks with an ALN," *1993 RSJ/IEEE International Conference on Intelligent Robots and Systems*, pp. 289 - 293, 1993.
- LaMoyne, et al, "Genetic Model Reference Adaptive Control," *IEEE International Symposium on Intelligent Control*, Columbus, Ohio, pp. 219 - 224, 1994.
- Ram, et al, "Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation," *Adaptive Behavior 2*(3), pp. 277 - 304, 1994.
- Real World Interface, Inc. Dublin, NH.
- R. Robertson and R. L. Riolo, "A Tale of Two Classifier Systems," *Machine Learning 3*, pp. 139 - 159, 1988.
- Sims, "Evolving Virtual Creatures," *SIGGRAPH '94 Proceedings*, pp. 15-22, July 1994.
- S. D. Whitehead and D. H. Ballard, "Active Perception and Reinforcement Learning," *Proc. 7th International Conference on Machine Learning*, B. Porter and R. Mooney, eds., pp. 179-188, 1990.
- W. Wilson, "Classifier Systems and the Animat Problem," *Machine Learning 2*, pp. 199-223, 1987.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE October 20, 1997	3. REPORT TYPE AND DATES COVERED final, June 1994 - September 1997		
4. TITLE AND SUBTITLE Research and Design of Intelligent Many-Agent Systems		5. FUNDING NUMBERS N00014-94-0676		
6. AUTHOR(S) John S. Bay				
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) Virginia Polytechnic Institute and State University Blacksburg, Virginia 24061		8. PERFORMING ORGANIZATION REPORT NUMBER FRS 430569		
9. SPONSORING / MONITORING AGENCY NAMES(S) AND ADDRESS(ES) Office of Naval Research 800 N. Quincy Street Arlington, VA 22217		10. SPONSORING / MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES				
a. DISTRIBUTION / AVAILABILITY STATEMENT		12. DISTRIBUTION CODE		
13. ABSTRACT (Maximum 200 words) <p>This report documents efforts under ONR grant no. N00014-94-1-0676. This is an AASERT award attached to parent grant NRL no. N00014-93-1-G022. The purpose of the grant is to support research on how group dynamics can emerge from collections of agents that would enable them to make decisions that individuals could not or accomplish tasks that individuals could not.</p> <p>Funding from the grant supported four graduate students directly; i.e., with stipends and tuition, and a number of undergraduate students indirectly, through materials and supplies purchases to support their independent study efforts in distributed intelligence and cooperative robotics.</p> <p>Results of these studies indicate that among distributed/cooperative learning methods, the most promising and appropriate for distributed mobile agent applications is a combination of learning and behavioral methods. In particular, the recommended method combines the data structures and execution cycle of the learning classifier system with reinforcement computed similarly to Q-learning and with some stochastic selection and genetics-based rule-parsing methods. These systems, in conjunction with message-based communications between agents, is shown to be widely applicable and convergent in ideal scenarios. The methods have the disadvantages of being slow, and they do not perform well in sequential learning tasks without significant modifications.</p>				
14. SUBJECT TERMS Distributed Artificial Intelligence, Machine Learning, Distributed Robotics			15. NUMBER OF PAGES 53	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT unclassified	20. LIMITATION OF ABSTRACT UL	

October 20, 1997

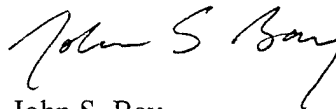
Dr. Michael Shneier
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5660

Dear Dr. Shneier:

Enclosed please find three copies of the final report for my project N00014-94-1-0676. It includes a brief summary of the work performed by the students supported under this AASERT project, and contains copies of the abstracts of their graduate theses. In addition you will find attached photocopies of all papers which these students authored, co-authored, or to which they otherwise contributed.

It has been a pleasure performing this work with ONR support. I hope we have served the Navy's needs. If I can be of further assistance, please do not hesitate to contact me.

Sincerely,



John S. Bay
Associate Professor

cc: Administrative Grants Officer (1 copy)
Director, Naval Research Laboratory (1 copy)
Defense Technical Information Center (2 copies)
ONR Code 11SP (1 copy)